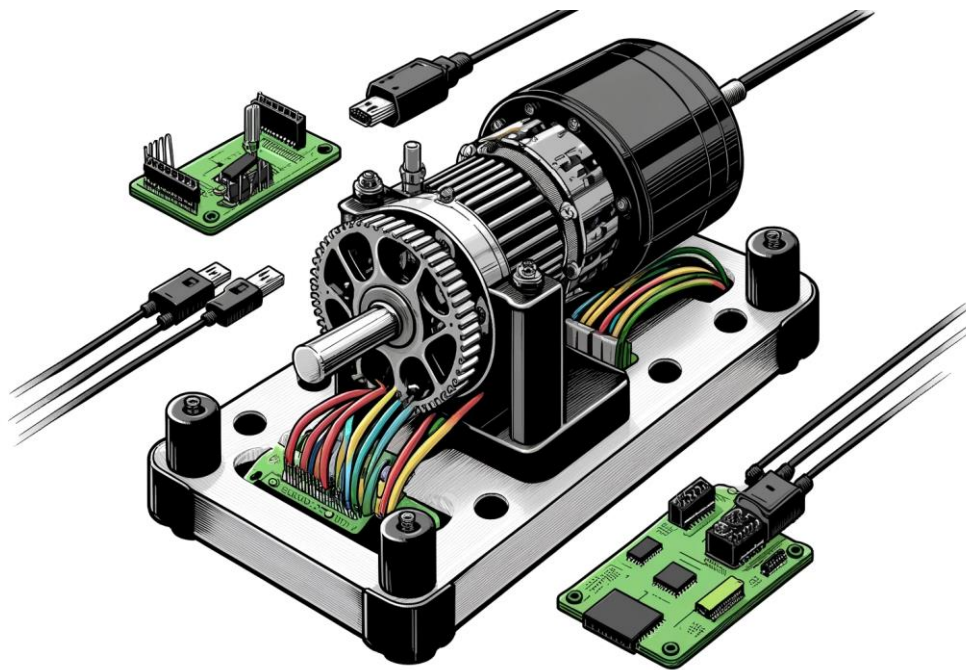


FMH606 Master's Thesis 2024

IIA

# Clustering Time-Series Data



Ref.: [1]

Anders Rudolfson

Faculty of Technology, Natural Sciences and Maritime Sciences

Campus Porsgrunn

**Course:** FMH606 Master's Thesis, 2024

**Title:** Clustering Time-Series Data

**Number of pages:** 53

**Keywords:** Classification, Clustering, LSTM, CWT, DTW, K-Means, K-Medoids, VGG19, ResNet18, PCA, Test Rig, Vibration Analysis, CBM

**Student:** Anders Rudolfson

**Supervisor:** Ole Magnus Brastein, Nils-Olav Skeie

**External partner:** **BEDKO AS**

**Summary:**

This thesis investigates the application of machine learning techniques to classify and cluster time-series data from motor vibrations, focusing on enhancing condition-based monitoring (CBM) systems for industrial automation.

The primary aims are to assess the data generation capabilities of a custom-built test rig and to evaluate the effectiveness of various classification and clustering methods on this data.

The study explores several techniques, including Dynamic Time Warping (DTW), Continuous Wavelet Transform (CWT), Long Short-Term Memory networks (LSTM), and convolutional neural networks such as ResNet18 and VGG19, paired with clustering algorithms like K-Means and K-Medoids.

The findings confirm that the test rig is effective for generating reliable data. While LSTM models excel in managing complex datasets with minimal variations, traditional classification methods like ResNet18 combined with CWT are highly accurate but struggle with subtler datasets. Clustering methods show promise but vary in effectiveness depending on the distinctiveness of the data. This research underscores the potential of advanced data processing techniques to improve the reliability and efficiency of CBM systems in industrial settings.

# Preface

This Master's thesis, titled "Clustering Time-Series Data," represents the culmination of my academic journey in the Master's program in Industrial IT and Automation at the University of South-Eastern Norway.

I am immensely grateful to my supervisor, Ole Magnus Brastein, and my co-supervisor, Nils-Olav Skeie, for their invaluable guidance, patience, and expertise throughout this project. Their insights and feedback have been crucial in shaping this research.

A special thank you goes to BEDKO AS, our external partner, who provided not only the test rig that was essential for my experiments but also continuous support throughout my research. This collaboration has been instrumental in bridging the gap between theoretical knowledge and practical application.

I would also like to express my gratitude to my family, whose endless support and encouragement have been my anchor throughout my studies. To my fellow students, who have been a source of motivation and camaraderie, thank you for all the stimulating discussions and for sharing this educational journey with me.

Lastly, I extend my appreciation to all the faculty members and administrative staff at the university who have contributed to my academic and personal growth during this program. This thesis would not have been possible without the supportive and enriching environment provided by the university.

Anders Rudolfsen

Porsgrunn, 15.05.2024

# Contents

Preface .....	3
Contents.....	4
1 Introduction .....	6
2 System Description.....	8
2.1 Test Rig.....	8
2.1.1 Accelerometer - MPU9250 .....	8
2.2 Data Acquisition Pipeline.....	9
2.2.1 Babel.....	9
2.2.2 InfluxDB.....	10
2.3 Datasets .....	10
2.3.1 Normal Operations .....	11
2.3.2 Vibrations.....	12
2.3.3 Vibrations Low.....	12
2.3.4 Friction .....	13
2.3.5 Friction Low.....	13
2.3.6 Ramp-up and Ramp-Down .....	14
3 Literature Review .....	17
4 Methods .....	19
4.1 Dynamic Time Warping (DTW) .....	19
4.2 Continuous Wavelet Transform (CWT) - Scalogram .....	20
4.3 Classification Methods.....	21
4.3.1 LSTM.....	21
4.3.2 ResNet18 .....	22
4.4 Clustering .....	22
4.4.1 K-Means .....	22
4.4.2 K-Medoids .....	23
4.4.3 PCA.....	23
4.4.4 VGG19 .....	23
4.5 Pre-processing.....	24
4.5.1 Sequencing.....	24
4.5.2 Normalization.....	24
4.5.3 Dataset balancing .....	24
4.5.4 Generating the Scalograms .....	25
4.6 Metrics .....	26
4.6.1 Confusion Matrix .....	26
4.6.2 Recall.....	27
4.6.3 Precision .....	27
4.6.4 F1 Score .....	28
4.6.5 Performance Metrics for Clustering.....	28
5 Result.....	30
5.1 Datasets .....	30
5.2 Classification.....	32
5.2.1 Classification using ResNet18 and CWT .....	32
5.2.2 LSTM.....	34
5.3 Clustering .....	36
5.3.1 Clustering using VGG19 and CTW .....	36

*5.3.2 DTW + K-Medoids*..... **38**

**5.4 Comparison** ..... **39**

*5.4.1 Classification comparison* ..... **39**

*5.4.2 Clustering comparison* ..... **42**

*5.4.3 Real-world implementation* ..... **44**

**6 Discussion** ..... **46**

    6.1 Classification Performance ..... **46**

    6.2 Clustering Performance ..... **46**

    6.3 Methodological Considerations ..... **47**

    6.4 Practical Implications ..... **47**

    6.5 Limitations and Challenges ..... **47**

    6.6 Future Work ..... **47**

**7 Conclusion** ..... **49**

**References** ..... **50**

**Appendices** ..... **53**

# 1 Introduction

In the landscape of industrial automation and monitoring, Condition Based Monitoring (CBM) has emerged as a pivotal strategy to enhance operational efficiency and system reliability[2]. CBM utilizes real-time data to pre-emptively identify potential malfunctions or inefficiencies within mechanical systems, thereby circumventing extensive downtime and costly repairs[2]. With the right certainty, a condition-based maintenance system can outperform traditional time-based maintenance systems [1]. A central component of this approach is the classification of time-series data, which allows for the categorization of operational states or identification of anomalous events that could signify system degradation.

Despite the richness of time-series data captured through Supervisory Control and Data Acquisition (SCADA) systems in industrial contexts, a significant challenge persists due to the absence of accompanying metadata or "ground truth" reference data[3]. For effective CBM, data from multiple sensors is necessary [2]. This limitation severely constrains the feasibility of employing supervised learning techniques, which depend heavily on pre-labelled datasets to model system behaviour accurately.

To tackle this challenge, this thesis looks to validate a test rig specifically designed for developing and testing models for motor-based vibration classification and clustering. Vibration data, being relatively easy to collect from most existing systems, serves as an ideal subject for this research. The primary objectives are twofold: firstly, to ascertain whether the test rig can generate data suitable for classification and clustering; and secondly, to evaluate various classification and clustering methods applicable to motor vibration data. Exploring both classification and clustering is essential, as comparing these methods' results offers deeper insights into the data and the efficacy of the techniques employed. This dual approach facilitates a comprehensive understanding of clustering methods by using classification outcomes as a benchmark. The classification methods are selected based on two key criteria: reliability, ensuring the method performs well with time-series data, and exploratory potential, aimed at investigating new and promising techniques. To select methods, a literature review is performed arriving at four different methods, two for classification and two for clustering.

The structure of this thesis is as follows: Chapter 1 introduces the background, problem statement, and objectives. Chapter 2 details the test rig setup, data collection pipeline, and the methodologies for conducting experiments to gather various datasets. Chapter 3 comprises the literature review, providing relevant context and background information. Chapter 4 outlines the different methods employed, including pre-processing, classification, clustering, and metrics. Chapter 5 presents and compares the results of the classification and clustering methods, along with real-world implementation and dataset analysis. Chapter 6 discusses the findings and suggests areas for future research, while Chapter 7 concludes the thesis.

The practical implications of this research are significant for industries relying on motor-based systems. By improving the reliability of CBM through better classification and clustering methods, companies can reduce downtime and maintenance costs. The theoretical contributions include advancing the understanding of time-series data classification in industrial contexts and exploring new methodologies for data analysis. The findings contribute to the field of CBM by validating the test rig as a viable tool for generating high-quality data for classification and clustering research. This thesis also provides insights into

the effectiveness of various methods, helping to guide future research and applications in industrial CBM systems.

## 2 System Description

This chapter examines the foundational elements of the thesis, including the physical test rig, the data acquisition system, and the various experimental setups employed for data collection. To aid in the evaluation of the machine learning methods, the physical test rig is used, as it creates the possibility of creating custom datasets for detection.

### 2.1 Test Rig

The test rig, illustrated in Figure 1, comprises several components: a motor, two accelerometers, a motor control board, and a central micro controller that manages data communication and sensor measurements. The motor is secured with two brackets; one supports the motor itself, and the other is affixed to the motor's shaft using a bearing. Each accelerometer is capable of measuring acceleration in the x, y, and z directions.

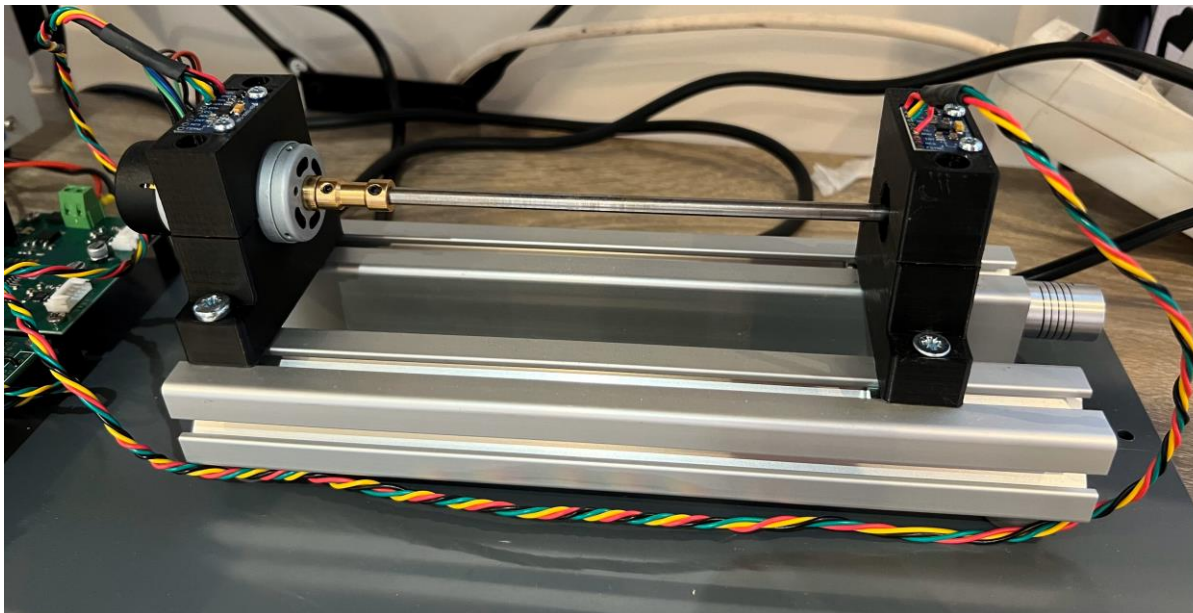


Figure 1: Image of the Test Rig

This rig has been provided by BEDKO AS [4] specifically for use in this thesis. BEDKO AS is responsible for supplying the rig, while the modifications and testing necessary for the experiments are conducted by the author. This setup forms the basis for collecting data essential for analyzing the performance and operational states of the motor under various conditions.

#### 2.1.1 Accelerometer - MPU9250

The MPU-9250 is a compact 9-axis Motion Processing Unit designed for consumer devices like smartphones and wearables[5]. It integrates a gyroscope, accelerometer, and digital compass into a tiny 3x3x1mm package[5]. This device offers improved noise performance and a broader compass range, with low-power modes to enhance battery life[5].



## 2.2 Data Acquisition Pipeline

To manage data acquisition and motor control, two software tools are utilized: Babel and InfluxDB. Babel, developed by BEDKO AS [4], is a graphical testing tool that facilitates motor control and displays sensor measurements. InfluxDB, is a database specifically designed for storing and providing real-time insights into time series data [6].

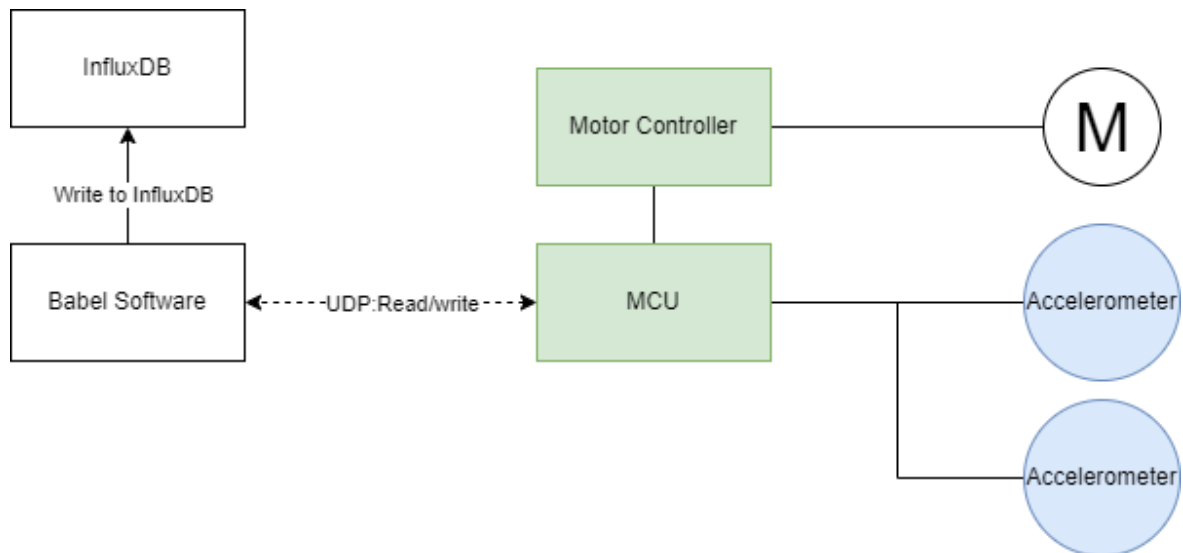


Figure 2: Test Rig Dataflow Diagram

The data flow, depicted in Figure 2, outlines the operations within the data acquisition pipeline. The process begins with Babel (2.2.1), which not only controls the motor but also provides monitoring capabilities for capturing data from the sensors. Babel interfaces with a MCU that processes inputs from the accelerometers and manages the motor's operations. From the MCU, the accelerometer data as well as additional data such as motor speed is sent back to Babel. From there Babel sends the data to Influx, where the data is stored. The datasets can subsequently be accessed from InfluxDB for analysis and further processing.

### 2.2.1 Babel

Babel is a versatile graphical testing tool that plays a critical role in the operation of the test rig used in this thesis[4]. Babel is a general testing tool, where dashboards can be created with custom functions[4]. For this thesis BEDKO AS provided a dashboard, depicted in Figure 3.

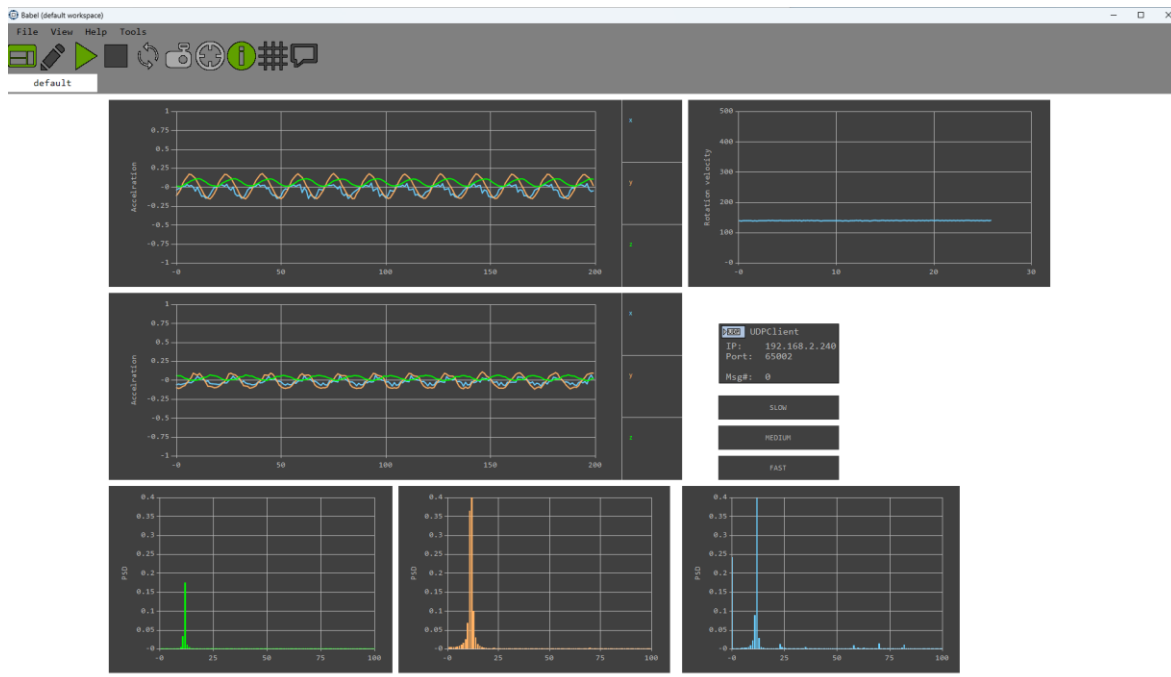


Figure 3: Babel Interface used for running experiments and creating datasets. The dashboard shows the two accelerometers, a chart of the speed as well as other useful information. There is also a possibility of selecting different speeds.

### 2.2.2 InfluxDB

InfluxDB is a robust time series platform that consolidates essential tools into a single binary, streamlining the management and analysis of time series data [7]. It features a unified API that simplifies data ingestion, querying, and storage, enhancing developer productivity [7]. Optimized for handling large volumes of time series data, InfluxDB supports high throughput and flexible data management, making it ideal for large-scale deployments[7]

## 2.3 Datasets

Utilizing the test rig and its associated acquisition pipeline, seven distinct datasets are collected, as detailed in Table 1. These datasets include Normal, Vibration, Vibration\_low, Friction, Friction\_low, RampUp, and RampDown, with further descriptions available in sections 2.3.1 through 2.3.6. The first five datasets; Normal, Vibration, Vibration\_low, Friction, and Friction\_low, are uniform in duration, each spanning 120 seconds and comprising 1000 measurements per second. Conversely, the RampUp and RampDown datasets vary in length, a variation attributable to the specific methods used for their collection.

Table 1: Datasets created through experiments


Name	Size	Length [s]
Normal [N]	120000x7	120
Vibration [V]	120000x7	120
Vibration_low [VL]	120000x7	120
Friction [F]	120000x7	120
Friction_low [FL]	120000x7	120
RampUp [RU]	171800x7	171.8
RampDown [RD]	68806x7	68.8

To gather the RampUp and RampDown data, a single dataset is initially produced by setting the motor speed to follow a saw-tooth pattern. This dataset is subsequently segmented into four parts: RampUp, RampDown, Bottom, and Peak. The Bottom and Peak sections are removed, as the focus of this dataset is on detecting changes in speed. A more detailed explanation of this process is provided in Chapter 2.3.6.

### 2.3.1 Normal Operations

The Normal Operations dataset serves as the foundational baseline for all measurements. This dataset is designed to represent typical operational conditions, providing a reference point for the CBM system. In practice, this is the state the classifier is expected to recognize as normal. Any deviations from this baseline are intended to be flagged as outliers or indications of potential issues. The methodology for collecting this data is outlined in Table 2.


Table 2: Normal Operations

Normal Operations
<p><b>Purpose:</b> The purpose of this experiment is to get a baseline dataset for normal operations. The experiments focus on running with no added disturbances.</p>
<p><b>Scope:</b> Normal Operations, Baseline</p>

<p><b>Setup:</b> The setup consists of the standard motor with no additional modifications and a functional bearing. The speeds to run the tests at should follow a predetermined plan.</p>
<p><b>Expectation:</b> Little to no vibrations</p>

### 2.3.2 Vibrations

The Vibrations dataset is designed to simulate a worst-case scenario, where detection of anomalies should be relatively straightforward. This experiment induces heavy vibrations in both the motor and shaft, creating a pronounced difference from normal operational conditions. The detailed steps of the experiment procedure are systematically outlined in Table 3.

Table 3: Operations with Vibrations


Operations with Vibrations
<p><b>Purpose:</b> The purpose of this experiment is to generate strong vibrations. The experiment is intended to be a worst case.</p>
<p><b>Scope:</b> Vibrations, Worst Case</p>
 <p>The diagram shows a horizontal shaft supported by two black vertical bearings. A grey cylindrical motor housing is on the left. A triangular object is balanced on top of the shaft between the bearings. A blue square is on top of each bearing. A thick black horizontal line is at the bottom.</p>
<p><b>Setup:</b> The experiment uses an object laying on top of the shaft, that will jump around when the shaft rotates, causing vibrations.</p>
<p><b>Expectation:</b> Strong vibrations</p>

### 2.3.3 Vibrations Low

The Vibrations Low dataset serves as a milder version of the Vibrations dataset. It is collected by applying a small piece of tape to create an uneven load on the motor, resulting in less intense vibrations. The methodology for collecting this dataset is detailed in Table 4.

Table 4: Operations with low Vibrations


Operations with low Vibrations
<p><b>Purpose:</b> The purpose of this experiment is to generate weak vibrations. The experiment is intended to cause a small amount of vibrations, to test the limit of classification.</p>
<p><b>Scope:</b> Vibrations</p>


<p><b>Setup:</b> This experiment is performed using a small bit of tape to create an uneven load, leading to vibrations.</p>
<p><b>Expectation:</b> Almost no vibrations. Should be similar to the baseline dataset.</p>

### 2.3.4 Friction

The Friction dataset is generated by inducing friction through a slight twisting of the bearing, which mimics friction within the bearing itself. This setup is intended to simulate a realistic scenario where the bearing has sustained some damage, leading to increased friction. This condition is representative of typical issues that might occur in industrial settings, providing valuable data for testing the effectiveness of condition monitoring systems. The complete procedure for creating this dataset is outlined in Table 5.


Table 5: Operations with Friction

Operations with Friction
<p><b>Purpose:</b> The purpose of this experiment is to create a dataset with quite high friction.</p>
<p><b>Scope:</b> Friction</p>

<p><b>Setup:</b> This experiment is performed by slightly rotating the right most support of the shaft to cause friction in the bearing.</p>
<p><b>Expectation:</b> High friction, reduced speed of the shaft</p>

### 2.3.5 Friction Low

The Friction Low dataset serves as a less intense version of the Friction dataset. Given the challenges of precisely adjusting the level of friction in the bearing as done in the original Friction dataset, an alternative method is employed to simulate a milder friction scenario. This involves using a sponge to press against the shaft, thereby creating slight friction within the system. This method offers a controlled way to emulate a lower degree of wear or damage. The full procedure for creating and capturing this dataset is detailed in Table 6.

Table 6: Operations with low Friction

Operations with low Friction
<p><b>Purpose:</b> The purpose of this experiment is to generate friction. The experiment is intended to cause a small amount of friction, to test the limit of classification.</p>
<p><b>Scope:</b> Bearing Wear</p>
 <p>The diagram shows a horizontal shaft supported by two black bearings. A green rectangular block is placed on a surface below the shaft, between the two bearings, to create friction.</p>
<p><b>Setup:</b> This experiment is performed by placing a sponge/soft object under the shaft, creating some amount of friction. The soft object should not be pressing too hard against the shaft.</p>
<p><b>Expectation:</b> Small amount of friction. Might cause some slowdown to the shaft speed, but will most likely be very similar to the baseline dataset</p>

### 2.3.6 Ramp-up and Ramp-Down

The Ramp-Up and Ramp-Down datasets are generated by initially creating a single dataset in which the speed of the motor is varied to follow a saw-tooth pattern. This modulation of the motor's speed allows for the capture of data reflecting both increasing and decreasing operational velocities, as illustrated in Figure 4.

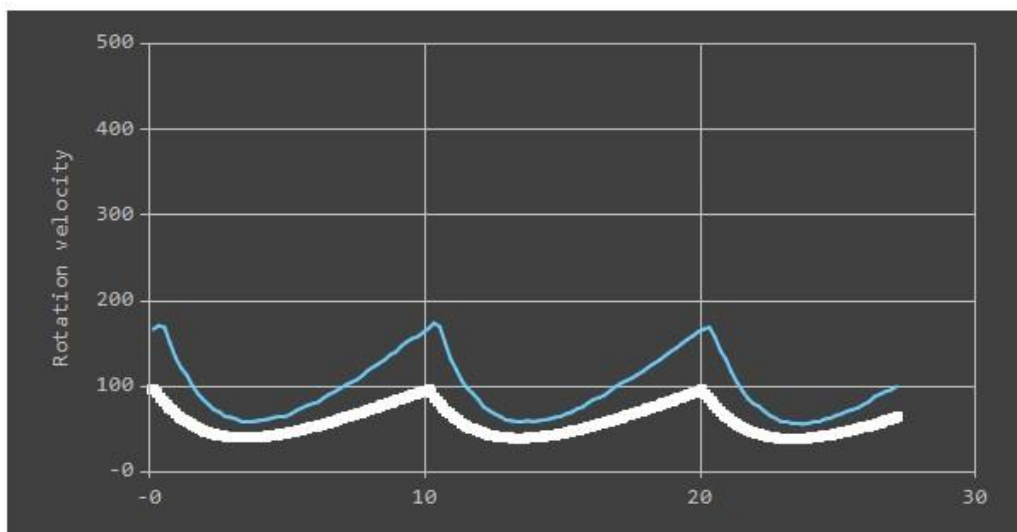


Figure 4: Input motor(white) and measured speed (Blue) taken from Babel. Note here that the input speed is not in the same unit as the Measured speed.

Data from the sensors, including the motor's speed, is collected to form the basis of the Ramp-Up and Ramp-Down datasets. The speed information is critical as it is used to segment the combined dataset into separate Ramp-Up and Ramp-Down phases. The process involves identifying the peaks (maximums) and bottoms (minimums) in the motor speed. Based on these points, the sections corresponding to Ramp-Up and Ramp-Down are extracted. However, areas immediately surrounding the peaks and troughs can be ambiguous and may inaccurately reflect either Ramp-Up or Ramp-Down characteristics. To address this, a small section around the maximums and a larger section around the minimums are deliberately excluded to ensure more accurate classification. This segmentation is visually represented in Figure 5, where the Ramp-Up phases are marked in orange and the Ramp-Down phases in red.

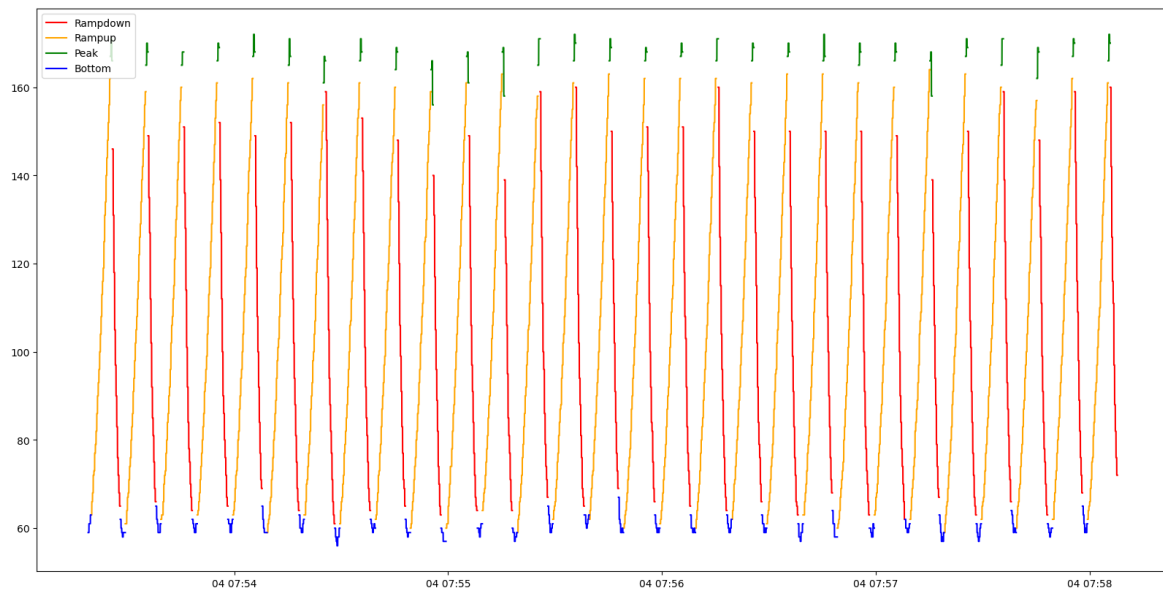
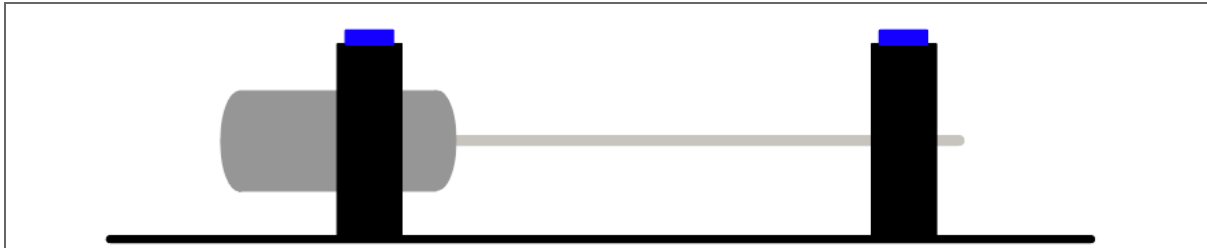


Figure 5: Ramp-Up and Ramp-Down extraction from original dataset

As depicted in Figure 4, the Ramp-Down section of the dataset is shorter than the Ramp-Up section, which accounts for the differences in their respective lengths as documented in Table 1. The physical setup of the test rig during these experiments remains consistent with the configuration used for the Normal Operations dataset, without any modifications. The specific setup details of the rig for these experiments are outlined in Table 7

Table 7: Operations with Ramp-Up and Ramp-Down

Operations with Ramp-Up and Ramp-Down
<b>Purpose:</b> The purpose of this experiment is to create two datasets, one for ramp-up and one for ramp-down
<b>Scope:</b> Ramp-up, Ramp-down



**Setup:** This experiment is performed on the baseline setup, where rotation velocity is adjusted in software. The rotation velocity (blue in the graph) is adjusted following a saw pattern. The white line indicates rotation speed percentage and is the setpoint. The data collected is at a later stage separated into two different datasets, one for ramp-up and one for ramp-down.

**Expectation:** Unsure

In summary, the Normal dataset serves as a baseline. For easy detection, the Vibration and Friction datasets are created, while the Vibration Low and Friction Low datasets are designed to be more challenging to classify or cluster. Additionally, the RampUp and RampDown datasets introduce a varying dimension, as these datasets will change in speed and vibrations over time as the motor ramps up or down, in contrast to the more consistent nature of the other datasets.



### 3 Literature Review

In the field of Condition-Based Monitoring (CBM), leveraging time-series data to predict and prevent mechanical failures has become increasingly important [2]. This literature review examines the relevant research on time-series data classification and clustering, highlighting key methodologies, their applications, and identifying gaps in current knowledge.

Condition-Based Monitoring (CBM) is a proactive maintenance strategy that relies on real-time data to monitor the condition of machinery and predict failures before they occur [2]. This approach aims to enhance operational efficiency and reduce downtime by detecting potential issues early [2]. A critical component of CBM is the analysis of time-series data collected from various sensors embedded in industrial equipment.

Previous studies have demonstrated the effectiveness of CBM systems in various applications [8], [9]. However, a significant challenge in implementing CBM systems is the lack of "ground truth" data, which are essential for supervised learning techniques [3]. This limitation has driven the need for alternative methods for data collection and analysis.

To address the issue of lacking "ground truth" data, it is proposed to use vibration data to classify different operational states. This approach has been proven to work [9], [10], [11]. However, the focus of this thesis is not to validate the effectiveness of vibration analysis but to demonstrate that the test rig created for testing classification and clustering methods is viable. This thesis explores whether the test rig can generate data suitable for these methods and evaluates its effectiveness by comparing the results of two classification and two clustering methods. The methods selected for evaluation are based on two criteria: interest and functionality.

The first classification method selected is the Continuous Wavelet Transform (CWT) combined with a deep learning model, specifically ResNet18 [12], [13]. This method was chosen for its unique approach to classification, which garnered interest from both the supervisors and the author. CWT transforms time-series data into a time-frequency representation, which can then be processed by deep learning models to identify patterns [13].

The second classification method is the Long Short-Term Memory (LSTM) network. LSTM networks are renowned for their ability to handle sequences of data and capture long-term dependencies [14], making them particularly suited for time-series classification tasks.

For clustering, the thesis evaluates the Dynamic Time Warping (DTW) method as a similarity measurement tool. While K-Means is a natural consideration for clustering, research suggests that K-Medoids is a more suitable candidate when using DTW [9]. DTW is a well-known tool for comparing two time-series, but it is not the most common method used in clustering or classification. This method was selected based on interest from both the supervisors and the author.

The second clustering method involves using CWT combined with VGG-19 [15] for feature extraction, followed by PCA and K-Means for clustering. This method leverages the powerful feature extraction capabilities of deep learning models and the clustering efficiency of K-Means, making it an interesting approach for analyzing time-series data.

Using the selected methods and the test rig, this thesis aims to evaluate the test rig as a viable tool for future classification and clustering research. Additionally, the thesis seeks to explore the effectiveness of these methods by comparing classification techniques with clustering

techniques. The goal is to determine how well the data can be clustered and to develop a robust clustering model suitable for the test rig. This evaluation will provide insights into the test rig's potential for generating high-quality data for various machine learning applications in Condition-Based Monitoring (CBM).

## 4 Methods

The method chapter covers the different classification and clustering methods used in this thesis as well as the different metrics, tool and feature extractions preformed to run the models.

### 4.1 Dynamic Time Warping (DTW)

Dynamic Time Warping (DTW) is a technique to compare sequences of time-series data[16]. The algorithm finds the overall cost of aligning the two series and is preferred over Euclidean distance measurements as the Euclidean distance measurements is very sensitive to variations in time[17]. An example of this could when comparing pronounced words, some people will take a longer time to pronounce the words while others will do it in a shorter amount of time. Here DTW will give a more accurate distance measurement and takes into account the shift in regard to the time axis[16].

The DTW algorithm starts with two inputs of size  $N$  and  $M$ , as shown in equation ( 1 )[16]. Equation ( 2 ) shows the cost matrix[16]. For initialization  $D_{i,0}$  and  $D_{0,j}$  for all values of  $i$  and  $j$  is set to infinity as shown in equation ( 3 )[16].

$$x_1: N \text{ and } y_1: M \quad (1)$$

$$D \in \mathbb{R}^{(N+1) \times (M+1)} \quad (2)$$

$$\begin{aligned} i = 1 \rightarrow N: D_{i,0} &= \infty \\ j = 1 \rightarrow M: D_{0,j} &= \infty \end{aligned} \quad (3)$$

$$D_{i,j} = d(x_i, y_j) + \min \begin{cases} D_{i-1,j-1} \\ D_{i-1,j} \\ D_{i,j-1} \end{cases}, \text{ where } d(x_i, y_i) = |x_i - y_i| \quad (4)$$

The cost matrix is then created using equation ( 4 )[16]. The cost matrix that is returned creates a grid of values, and the optimal path to align the two series as well as the cost of the alignment can be found in the cost matrix[16]. To find the optimal alignment, start at  $D_{N,M}$  and trace back to  $D_{0,0}$ [16]. This is done by looking at each “square” in the matrix, find the lowest value of the other “squares” next to the “square” that is currently being looked at, and that has previously not been traveled, and trace the path all the way back to  $D_{0,0}$ [16]. An example of this is shown in Figure 6, where the infinity edges are removed. To find the cost of alignment, one can look at the “last” value of the cost matrix, which is  $D_{N,M}$ . This cost of alignment is very useful when comparing time-series data as it represents the difference between the two time-series[16].

(5

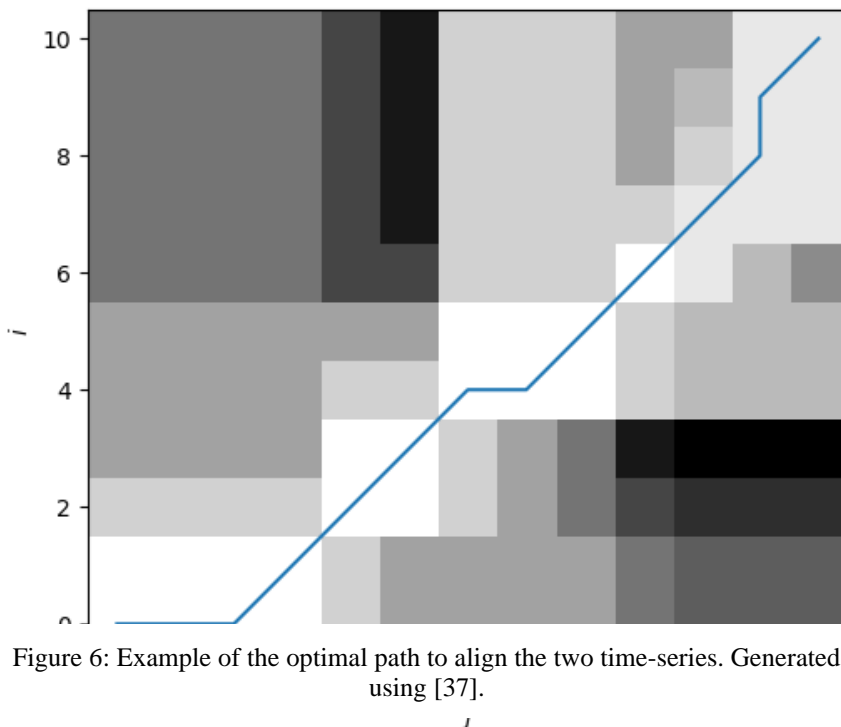


Figure 6: Example of the optimal path to align the two time-series. Generated using [37].

## 4.2 Continuous Wavelet Transform (CWT) - Scalogram

The Continuous Wavelet Transform (CWT) is a tool in signal processing that analyzes signals at different scales and resolutions[18]. The process begins with selecting a wavelet function, known as the mother wavelet, which effectively captures the signal's characteristics[18]. This wavelet is then scaled and shifted across the signal. Scaling adjusts the frequency content of the wavelet, enabling the analysis of different frequency bands, while shifting allows the examination of different parts of the time series[18].

The wavelet coefficients can be compiled into a two-dimensional graph called a scalogram (shown in Figure 7), where the x-axis represents time, the y-axis represents scale, and the intensity of the coefficients is often shown using color[19]. This visualization helps identify where various features occur in time and at what scales[19].

High coefficients indicate a strong match between the wavelet and the signal, revealing features such as spikes, breaks, or other characteristics at different times and scales[19]. CWT is especially useful for analyzing non-stationary signals, where frequency components change over time, such as in audio signals, geophysical data, or heart rate variability analysis [20], [21], [22]. It provides a detailed and intuitive way to examine these signals, uncovering insights that might be missed with other methods like the Fourier transform.

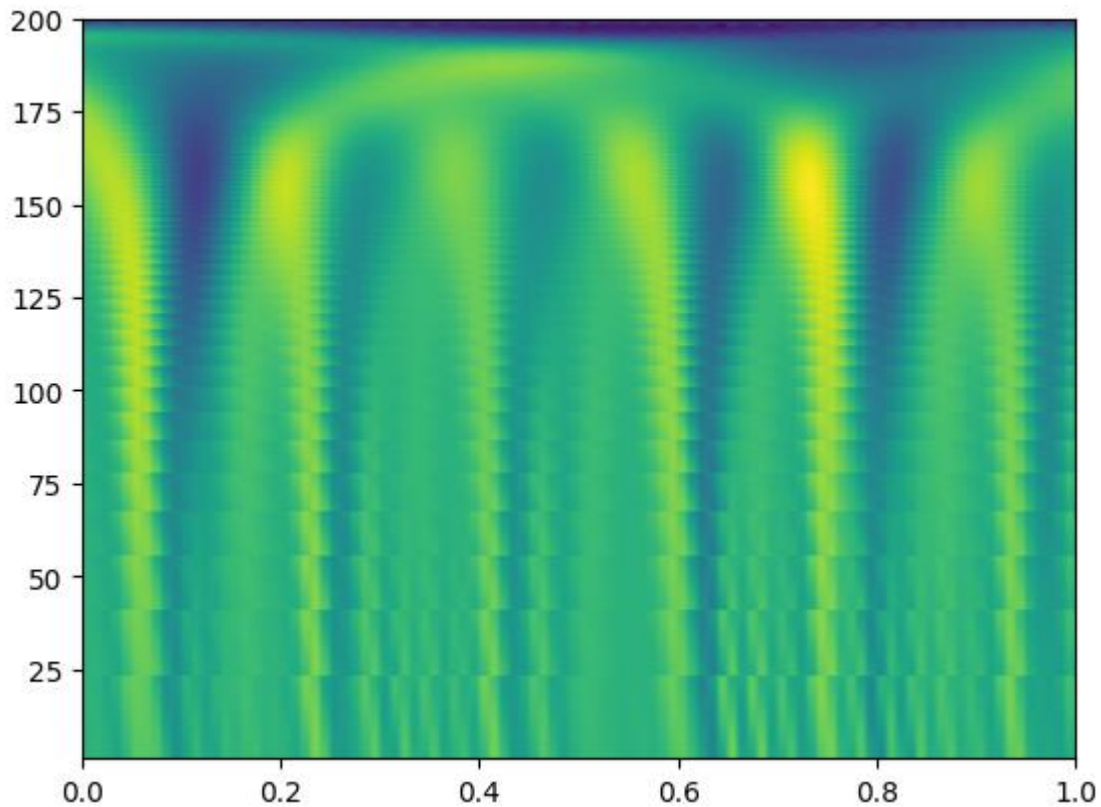


Figure 7: Example of scalogram. Data in scalogram taken from one sequence in Vibration data set

## 4.3 Classification Methods

This thesis covers two different classification methods, LSTM and ResNet18 using CWT scalograms. This chapter covers the two classification methods used.

### 4.3.1 LSTM

An LSTM cell has two states: a short-term state  $h$  and a long-term state  $c$  [14]. The cell uses various gates to control the flow of information, which are the forget gate, the input gate, and the output gate [14].

The forget gate determines which parts of the long-term state should be erased [14]. This gate is crucial for preventing the cell from being overwhelmed with irrelevant information [14].

The input gate controls which parts of the new input should be added to the long-term state [14]. This allows the cell to update its memory with new information that is deemed important [14]. Finally, the output gate decides which parts of the long-term state should influence the short-term state and be passed to the next layer [14]. This gate ensures that only relevant information is outputted at each time step [14].

These gates work together to manage the cell's memory effectively [14]. The forget gate first determines which information to discard from the long-term state [14]. Then, the input gate decides which new information to add [14]. After updating the long-term state, the output gate filters the updated state to produce the short-term state and the cell's output [14].

This gating mechanism allows LSTM cells to learn to recognize important inputs, store them in the long-term state, preserve them for as long as necessary, and extract them when needed[14]. This explains why LSTM cells have been so successful at capturing long-term patterns in various types of sequential data, such as time series, long texts, and audio recordings[14].

The equations that govern the operations within an LSTM cell are as follows: the input gate, forget gate, and output gate each have their own weight matrices and bias terms[14]. The current input vector and the previous short-term state are fed into these gates[14]. The forget gate's output is then multiplied element-wise with the previous long-term state to determine what to retain[14]. The input gate's output is multiplied with a newly computed candidate state to determine what new information to add to the long-term state[14]. Finally, the long-term state is transformed through the tanh function, and this transformed state is filtered by the output gate to produce the short-term state and the output[14].

By employing these techniques and structures, LSTM cells can effectively handle long sequences, capturing long-term dependencies in data such as time series, text, and audio[14].

### 4.3.2 ResNet18

ResNet-18 is an 18-layer deep convolutional neural network[12]. This pretrained model can identify and classify images into 1000 categories, such as keyboards, mice, pencils, and various animals[12]. Consequently, ResNet-18 has acquired detailed feature representations for a wide array of images. The network processes images with an input size of 224 by 224 pixels[12].

## 4.4 Clustering

In this thesis, two distinct clustering techniques are utilized: K-Means in conjunction with CWT, PCA, and VGG19, and K-Medoids paired with DTW. This chapter explores these clustering methods and the complementary tools employed to execute the clustering process.

### 4.4.1 K-Means

In this thesis, the K-means clustering algorithm is employed for its effectiveness in segmenting a dataset into K distinct clusters by minimizing the within-cluster sum of squares. This method is particularly favored for its simplicity and efficiency in large-scale data applications, such as pattern recognition and image segmentation.

The process begins with the random selection of K centroids, followed by the assignment of each data point to the nearest centroid. The centroids are recalculated iteratively until their positions stabilize, signifying convergence [23]. A key aspect of implementing K-means is determining the optimal number of clusters, K, often assessed using the Elbow method. This technique identifies a point where increasing the number of clusters ceases to yield significant improvements in variance reduction [24].

Data standardization is critical prior to applying K-means to prevent scale differences from distorting the clustering process[25]. The algorithm's sensitivity to the initial placement of centroids is addressed by multiple initializations, selecting the configuration that minimizes the within-cluster sum of squares for robustness [25].

Finally, the cluster validity is typically assessed using the silhouette score, which evaluates the compactness and separation of the clusters, ensuring that the clustering results are both meaningful and distinct.

#### 4.4.2 K-Medoids

When selecting a method to be used for with DTW, a natural thought is K-Means. Unfortunately, K-Means is not suited to be used with DTW[26]. Instead, K-Medoids is suggested as an alternative to K-Means when using DTW[26].

K-Medoids differ in the selection of data points. In K-Means uses the mean of data points, a centroid, while medoids is taken from the dataset itself. The most well-known implementation of medoid based algorithms is the PAM algorithm[27], which consists of three steps:

- **Initialization:** Choose  $k$  initial medoids randomly from the dataset.
- **Assignment Step:** Assign each data point  $x_j$  to the nearest medoid.
- **Update Step:** For each cluster update the medoid to the data point within the cluster that minimizes the total dissimilarity to the other points in the cluster.

The steps are then iterated on.

In the standard version of K-Medoid[27] Euclidean distance is used to measure between the points. In this thesis the only difference is the use of DTW as the “distance” measurement.

#### 4.4.3 PCA

Principal Component Analysis (PCA) is a statistical technique used for reducing the dimensionality of datasets while preserving as much variability as possible [28]. It transforms a set of possibly correlated variables into a smaller set of uncorrelated variables called principal components [28]. The first principal component captures the greatest amount of variance in the data, with each subsequent component accounting for as much of the remaining variability as possible [28].

PCA is widely applied in various fields such as technology, where it helps in optimizing engineering designs and processes by synthesizing multiple variables and measurements[29], and in chemometrics for analyzing large datasets typical in chemistry[30].

In summary, PCA is a versatile tool that aids in data simplification and analysis across multiple scientific and industrial applications by effectively reducing the number of variables to consider while retaining essential information.

#### 4.4.4 VGG19

VGG-19 is a deep convolutional neural network comprising 19 layers [15]. This pre-trained model can recognize and classify images into 1000 different categories, such as keyboards, mice, pencils, and a variety of animals[15]. As a result, VGG-19 has developed sophisticated feature representations across a broad spectrum of images[15]. The network processes images with an input size of 224 by 224 pixels[15].

## 4.5 Pre-processing

This chapter covers the different pre-processing steps that are used before performing classification and clustering.

### 4.5.1 Sequencing

The datasets extracted from the test rig (2.1) are continuous time series. When utilizing these datasets for classification and clustering methods, sequencing is essential [31]. In this thesis, two types of sequencing are employed: non-overlapping and overlapping sequencing [31].

A non-overlapping sequence with a length of 3 is structured as follows, as shown in equation ( 8 )[31]:

$$[t_1, t_2, t_3, t_4, t_5, t_6] \rightarrow [t_1, t_2, t_3], [t_4, t_5, t_6] \quad (6)$$

Conversely, an overlapping sequence with a length of 3 and an overlap of 2 is structured as follows, as shown in equation ( 9 )[31]:

$$[t_1, t_2, t_3, t_4, t_5, t_6] \rightarrow [t_1, t_2, t_3], [t_2, t_3, t_4], [t_3, t_4, t_5], [t_4, t_5, t_6] \quad (7)$$

### 4.5.2 Normalization

Normalization in machine learning refers to the process of scaling or transforming data to ensure that each feature contributes equally, enhancing the overall data quality and the performance of machine learning algorithms[32]. It is essential because raw data often contains variables that vary in scale and units, which can distort the predictive model if not standardized [33].

Normalization helps machine learning models to converge faster and perform better, as it prevents certain features from disproportionately influencing the model's output due to their range [34].For instance, it ensures that features with larger numerical ranges do not dominate those with smaller numerical values, allowing the model to learn more effectively from all features equally [35]

In summary, normalization is a crucial pre-processing step in machine learning that enhances model accuracy and stability by standardizing the scale of the data features, making the training process more efficient and effective.

### 4.5.3 Dataset balancing

When classifying or clustering, the balance of the dataset is crucial[36]. A dataset with disproportionate label sizes can lead to problems in prediction accuracy[36]. The definitions of mild, moderate, or extreme imbalance vary, but generally follow this guideline:

Table 8: Degree of imbalance in dataset. Taken from [36]

DEGREE OF IMBALANCE	PROPORTION OF MINORITY CLASS
---------------------	------------------------------



<b>MILD</b>	20-40% of data
<b>MODERATE</b>	1-20% of data
<b>EXTREME</b>	<1%

Comparing the degree of imbalance guidelines in Table 8 to the datasets in 2.3, the most imbalanced dataset is Ramp Down at 57.3%.

#### 4.5.4 Generating the Scalograms

Two of the methods in this thesis uses scalograms, the visual representation of CWT, in classification and clustering. A scalogram is a 2-D representation of a 1-D signal. This creates a problem, as there is no direct way of visualizing the six (X, Y, Z for the motor and for the bearing) 1-D signals being recorded.

There are multiple ways this can be solved. The data can be combined to create a new signal, or, some of the measurements can be discarded. Either way, those solutions are not ideal as data will be lost. As these methods rely on image detection, the scalogram itself is not important, as it's the image that is being created that's being used for classification and clustering. Therefore, an alternative approach to keeps as much data as possible is used.

This strategy starts with generation of a scalogram for each of the six measurements. The scalograms are the combined to make one big image, as shown in Figure 8. This allows all information to be kept.

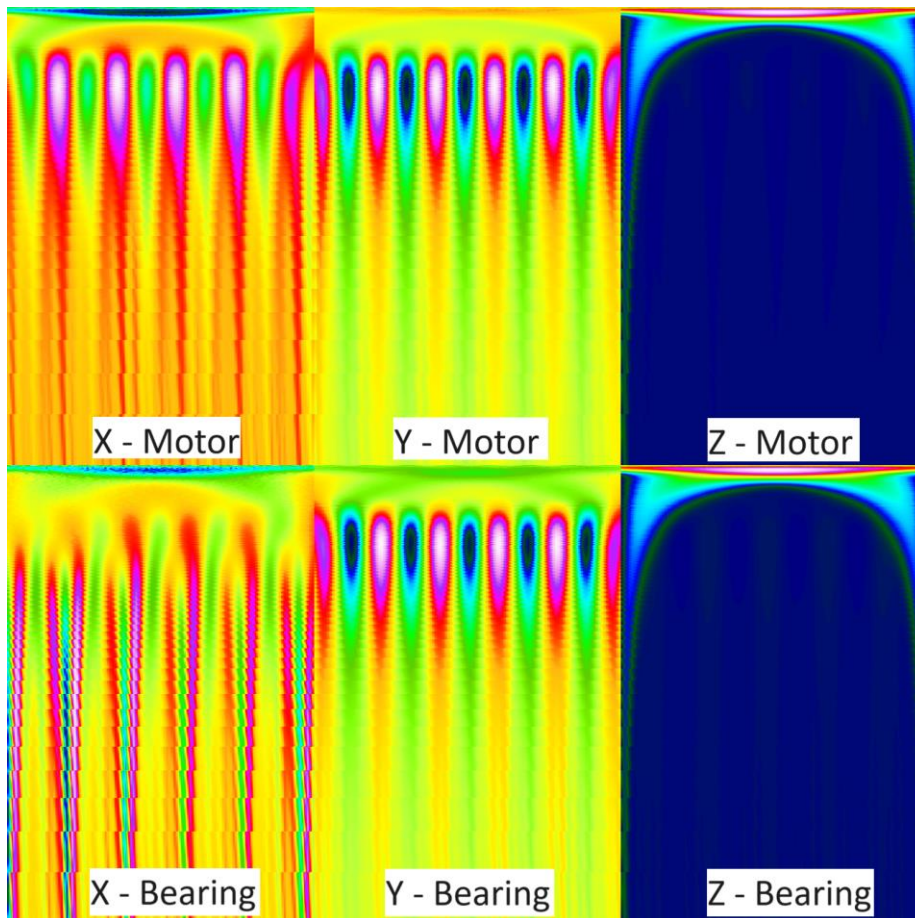


Figure 8: The full image consists of 6 scalograms, each from one measurement. This image shows which scalogram belongs to which measurement.

## 4.6 Metrics

When building a machine learning model, one of the most important components is the metrics used to evaluate the results. It's important to have metrics that well define the goal of the model, and accurately measures the difference between iterations.

### 4.6.1 Confusion Matrix

A confusion matrix is a table that displays the performance of a classification model on a specific dataset[14]. The dimensions of the matrix correspond to the number of classes[14]; for example, three classes would result in a 3x3 matrix, as shown in Figure 9. In binary classification models, however, the matrix is always 2x2. The primary purpose of the confusion matrix is to extract metrics like True Positives, True Negatives, False Positives, and False Negatives[14]. These metrics are essential for computing common classification metrics such as Recall and Precision[14].

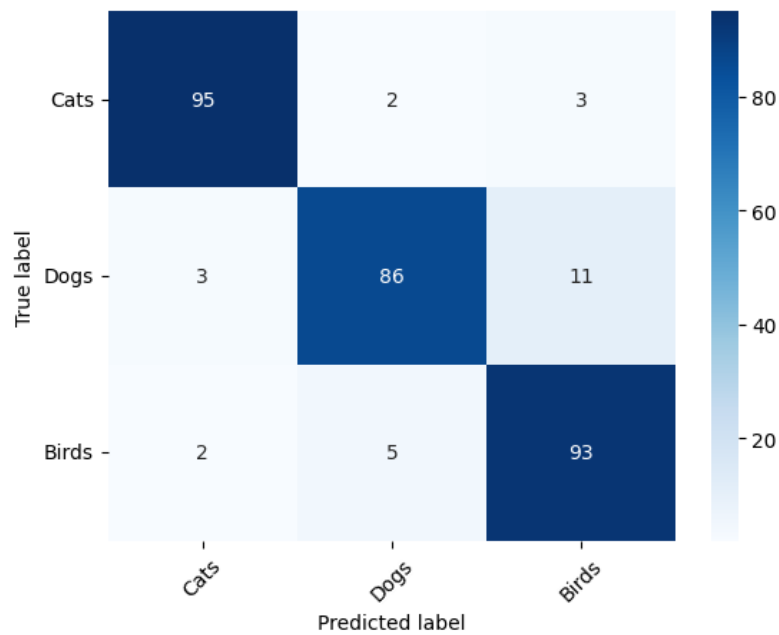


Figure 9: Multiclass Confusion Matrix

The confusion matrix is also valuable on its own as it clearly illustrates the errors a classification model makes[14]. For instance, Figure 9 shows an example where the model incorrectly identified 11 dogs as birds.

#### 4.6.2 Recall

Recall, also known as sensitivity, measures a model's ability to correctly identify its target[14]. It is defined as the ratio of true positives to the total of true positives and false negatives[14]. This metric is particularly valuable in scenarios where failing to detect true positives is critical, such as in medical diagnostics[14]. The calculation for recall is outlined in Equation ( 8 ).

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (8)$$

#### 4.6.3 Precision

Precision assesses a model's accuracy in predicting positive instances of the objective[14]. It is calculated as the ratio of true positives to all positive predictions made[14]. This metric is particularly important in situations where false positives carry significant consequences, such as in industrial environments where incorrect shutdowns can be costly[14]. The formula for precision is detailed in the Equation ( 9 ).

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (9)$$

#### 4.6.4 F1 Score

The F1 score measures a model's accuracy by balancing recall and precision[14]. A high F1 score suggests that a model achieves both high recall and precision[14]. This metric is particularly valuable when it's crucial to maintain a balance between recall and precision[14]. The formula for calculating the F1 score is presented in Equation ( 10 ).

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (10)$$

#### 4.6.5 Performance Metrics for Clustering

This thesis operates under the assumption that the number of clusters (n-clusters) is known, and the ground truth of the dataset in use is available, leading to an unconventional method for measuring performance. The primary objective is to compare clustering methods against classification methods. To this end, performance measurement of the clustering methods is depicted in Figure 10.

Figure 10 illustrates an ideal scenario where each animal is perfectly segregated into separate clusters without any class overlap. For example, all cats are consistently grouped into Cluster 1, demonstrating effective clustering performance. This pattern holds true for dogs and birds as well, highlighting the clustering method's strong performance.

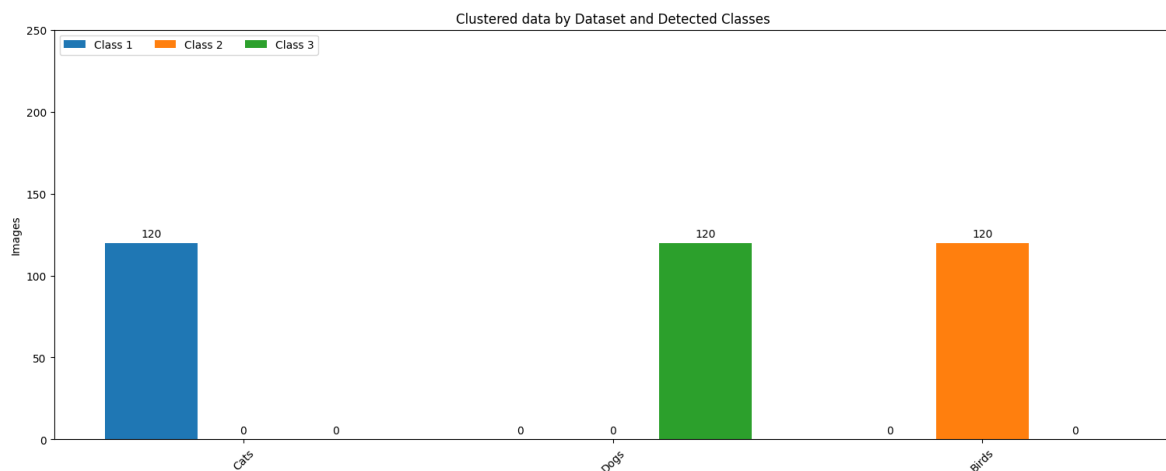


Figure 10: Clustering Performance Example 1

The second example, illustrated in Figure 11, depicts a model that struggles to differentiate between cats and dogs. While it successfully separates the birds, it lacks the confidence to accurately classify whether an animal is a cat or a dog.

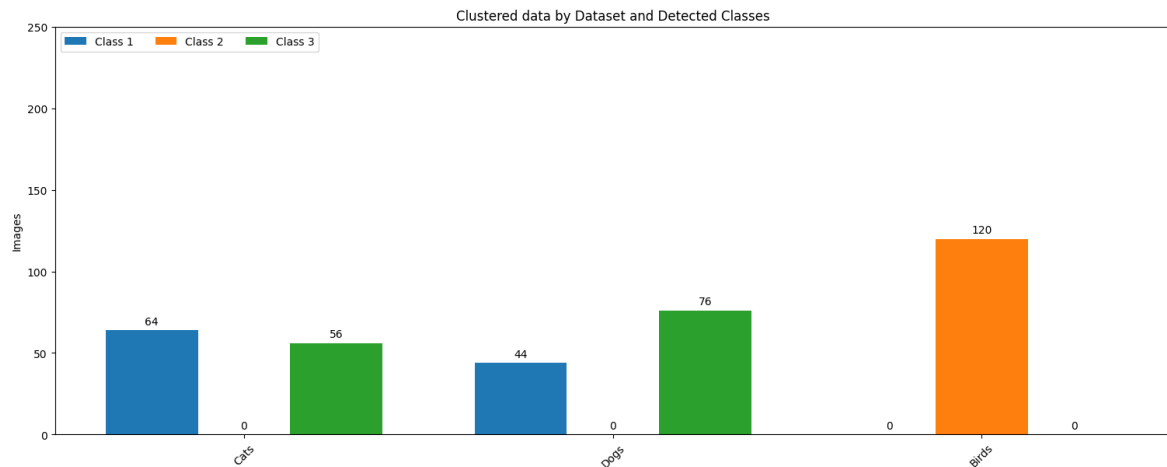


Figure 11: Clustering Performance Example 2

The bar charts provide a clear indication of the clustering methods' performance by showing the distribution of different classes across datasets. A high concentration of one class in a single dataset suggests that the model is performing well on that dataset. Conversely, if a class is dispersed across two or more datasets, it indicates that the model has difficulty distinguishing between the datasets. Ideally, a perfect clustering method would result in each class being exclusively concentrated in one dataset, as demonstrated in Figure 10.

## 5 Result

The result describes the results of running the different methods shown in chapter 4 on the datasets generated from the test rig, described in chapter 0. The results chapter goes through the results of experiments, the results of the different methods, both individually and compared to each other, and it covers how well the different methods are suited for real-time implementation.

### 5.1 Datasets

To develop a robust understanding of the models and set appropriate expectations, it is crucial to familiarize oneself with the various datasets and their comparative characteristics. As noted in section 2.3, by design the three most distinct datasets identified should be "normal" (N), "vibration" (V), and "friction" (F). Here, V and F represent the extremes, while the N dataset serves as the baseline. In contrast, the "vibration low" (VL) and "friction low" (FL) datasets are designed to be more challenging to distinguish since their deviations from the N conditions are minimal. The "ramp-up/down" (RU/RD) datasets are somewhat ambiguous, but they are expected to exhibit time-dependent characteristics as the RPM increases or decreases, which should be evident in the data.

A good way to visualize the difference between the different datasets is a scalogram. The scalograms as shown in Figure 12, are created with a sequence length of 120, where each image contains a scalogram for each measurement done (6 in total).

Figure 12 shows the first comparison between the N, V and F datasets. As expected, there is a clear visual difference between the datasets. The middle image shows the V dataset. Here it's clear that the lines are more sporadic compared to the two other datasets. Looking at the difference between N and F their shapes follow more of a similar pattern, but it's clear that there are more lines on the F. Overall there is a clear visual difference between the three datasets, as intended.

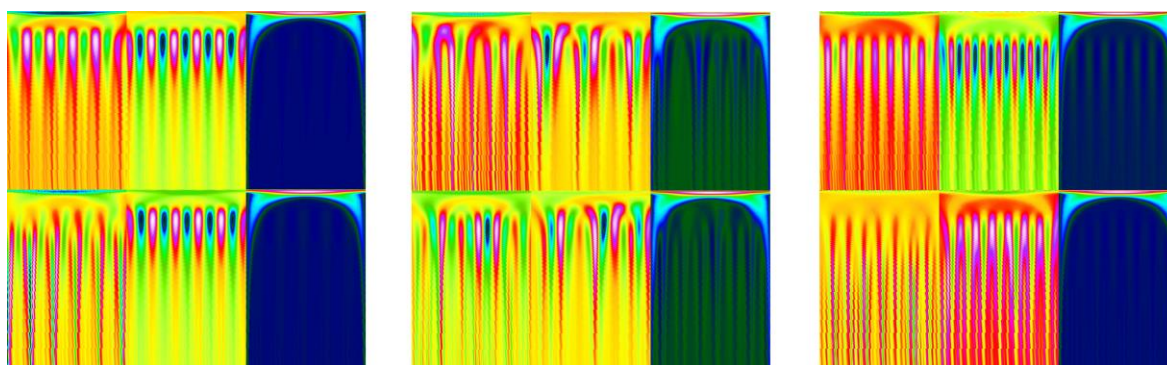


Figure 12: Comparison of scalograms sequence from left to right: Normal, Vibration, Friction

The next comparison is between the N, VL, and FL, shown in Figure 13. Here the difference is much less clear, especially between VL and FL. The N dataset is possible two distinguish between the two, but it's still less obvious than the above example.

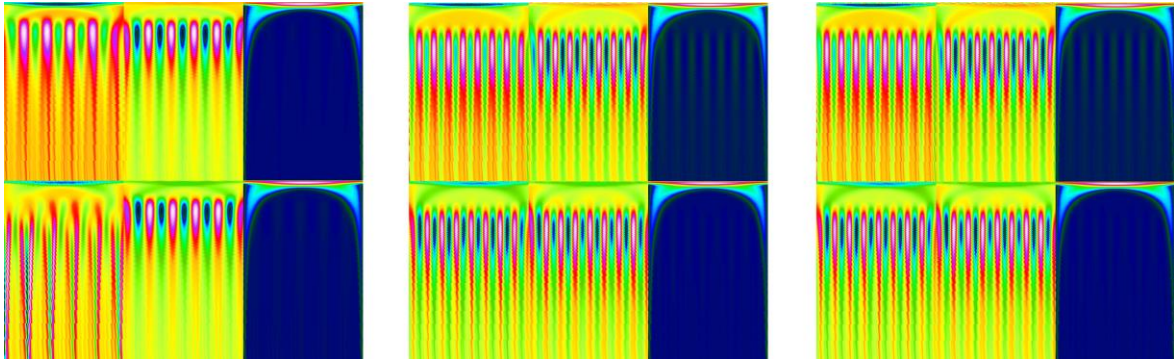


Figure 13: Comparison of scalogram sequences from left to right: Normal, Vibration Low, Friction Low

The last two datasets are the RU and RD datasets. Looking first at the RU dataset, shown in Figure 14. The image shows three sequences of data taken in order, showing sequences from one ramp up in speed. From the scalograms one can see the change in RPM, where the left most image shows fewer vertical lines than the right most image. While the first two images are quite distinct from the other datasets visually, the last one do look similar to the baseline.

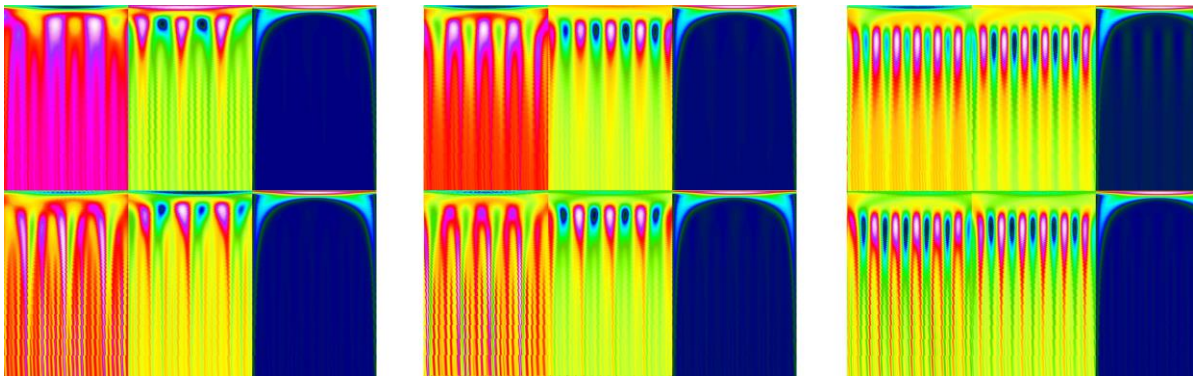


Figure 14: Scalograms of Ramp Up data. From left to right the scalograms are ordered in time, where the left most is at a lower RPM while the right most is at a higher RPM.

The RD data, depicted in Figure 15, is almost visually identical to the RD dataset with the exception of that the order of the images is reversed.

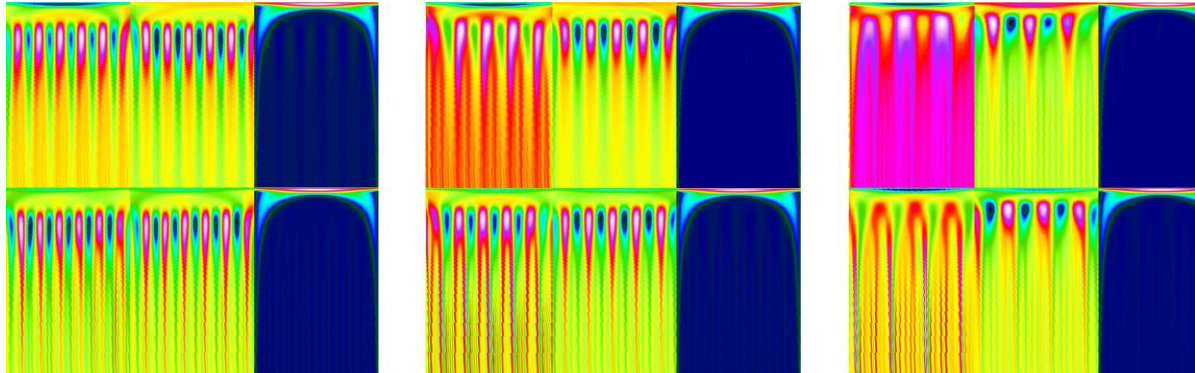


Figure 15: Scalograms of Ramp Down data. From left to right the scalograms are ordered in time, where the left most is at a higher RPM while the right most is at a lower RPM.

In summary, N, V and F are distinguishable as intended while VL and FL are similar, also as intended. The RU and RD datasets are not distinguishable when looking at them individually, but when set in context the difference between RU and RD is clear.

## 5.2 Classification

The classification results show how the classification methods were implemented, and the general results of the classification on the test sets.

### 5.2.1 Classification using ResNet18 and CWT

The classification model utilizing Continuous Wavelet Transform (CWT) is initiated by generating images as outlined in section 4.5.4. The process, outlined in Figure 16, begins with the creation of scalograms. These scalograms are subsequently divided into a training dataset and a testing dataset, 80% and 20% respectively. The next stage employs the ResNet18 model as a baseline. This model is then specifically trained on the training dataset to improve its ability to predict on the vibration data in the form of scalograms. Once the model is trained, the new model is then applied to the testing dataset to make predictions. Finally, performance metrics are derived from the predictions on the testing dataset, which is exclusively reserved for testing and never used in the training phase.

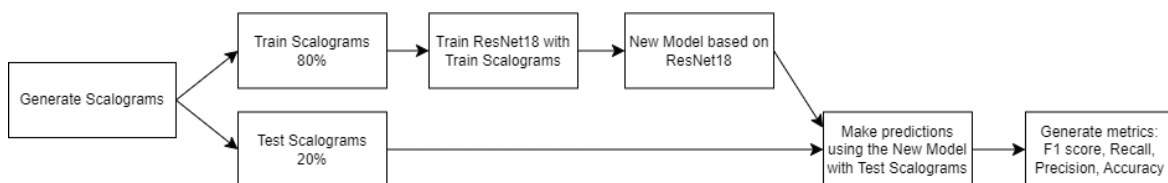


Figure 16: Overview diagram of ResNet18 using CWT images

To get a good understanding of how the model can classify the different datasets, different combinations of datasets are run. The first combination of datasets are the three most distinct datasets: N, V and F. The results are shown in Figure 17, where it's clear that the classification model works very well. All datasets are identified correctly.



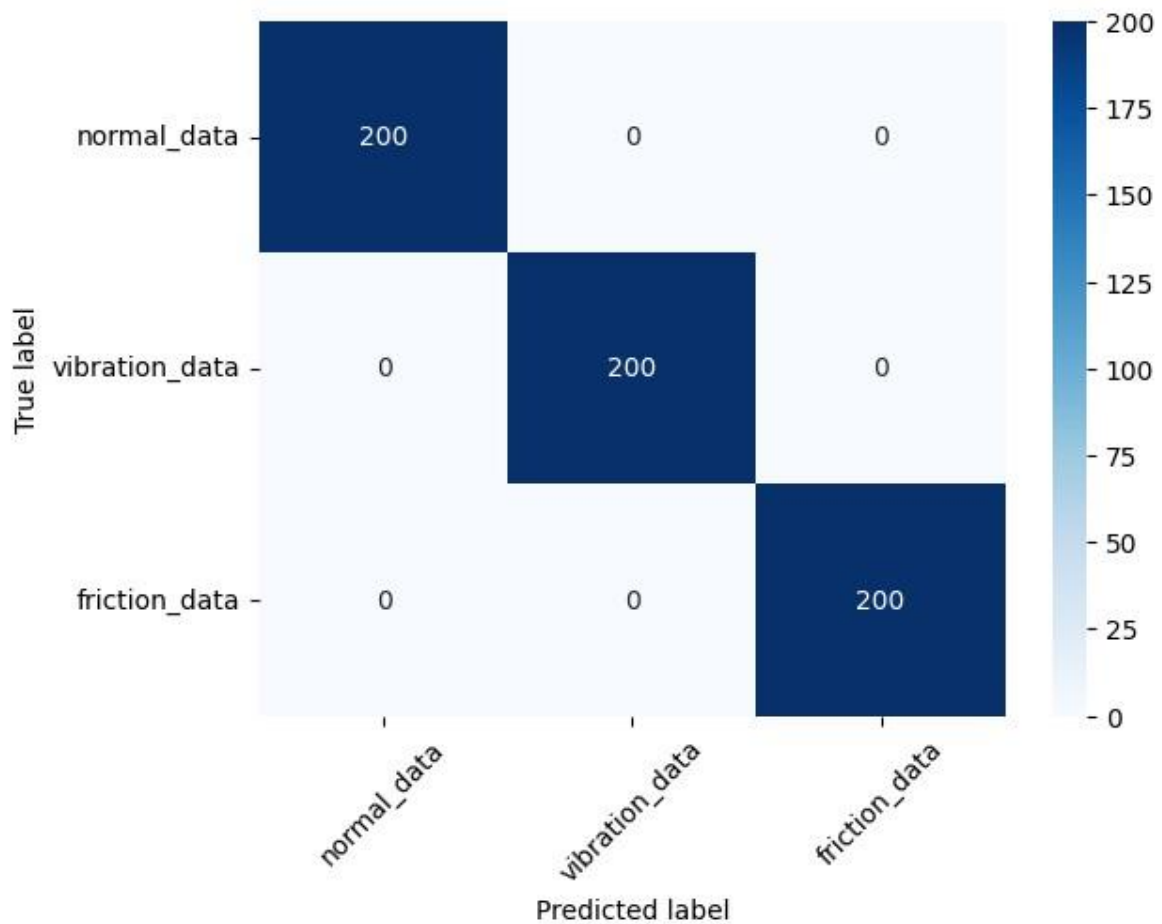


Figure 17: CTW Classification ([N,V,F], 15 epochs, lr0.001, seqLen120, batchSize32)

The next results are shown in Figure 18. The results depict five variables being used in the training and the results show the test data on the model. Here N, V and F shows the same results as the previous results, while the RU and RD data is not classified well.

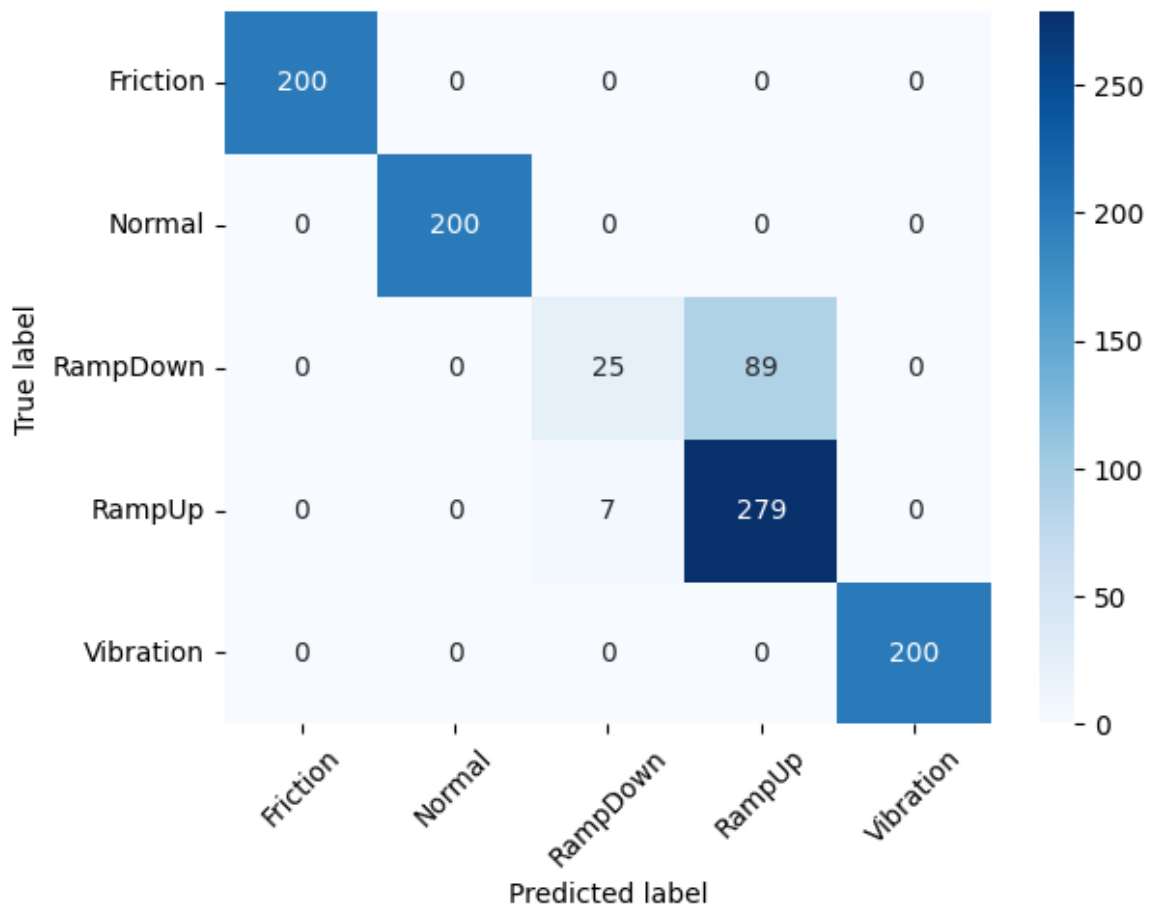


Figure 18: CTW Classification ([N,V,F,RU,RD], 15 epochs, lr0.001, seqLen120, batchSize32)

### 5.2.2 LSTM

The classification model leveraging Long Short-Term Memory (LSTM) networks commences by generating sequences as described in section 4.5.1. The comprehensive LSTM training and testing procedures are depicted in Figure 19. Initially, sequences are generated and subsequently divided into training, validation, and test sets at proportions of 60%, 20%, and 20%, respectively. In the following stage, the LSTM model is trained using both the training and validation datasets, with the latter serving to validate the training process. Upon completing the training, a new model is configured. This model then employs the test set to make predictions. In the final step, metrics are derived based on these predictions to assess the model's performance.

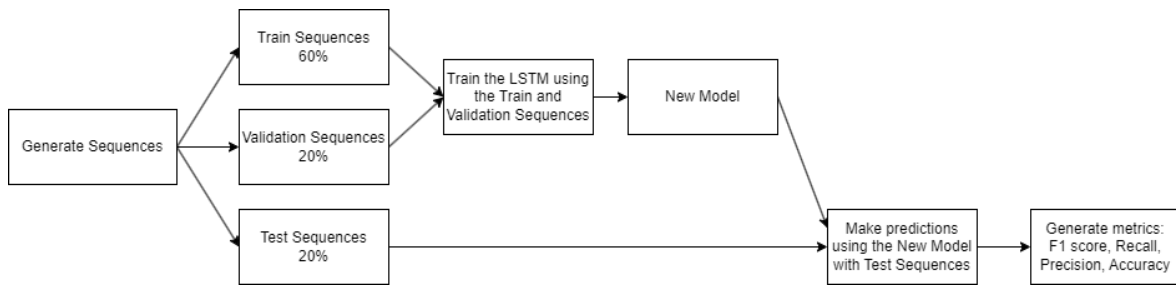


Figure 19: Overview diagram of LSTM training and testing process

The first combination of datasets is N, V and F, as shown in Figure 20. Here the LSTM classifier is doing a good job of classifying the datasets.

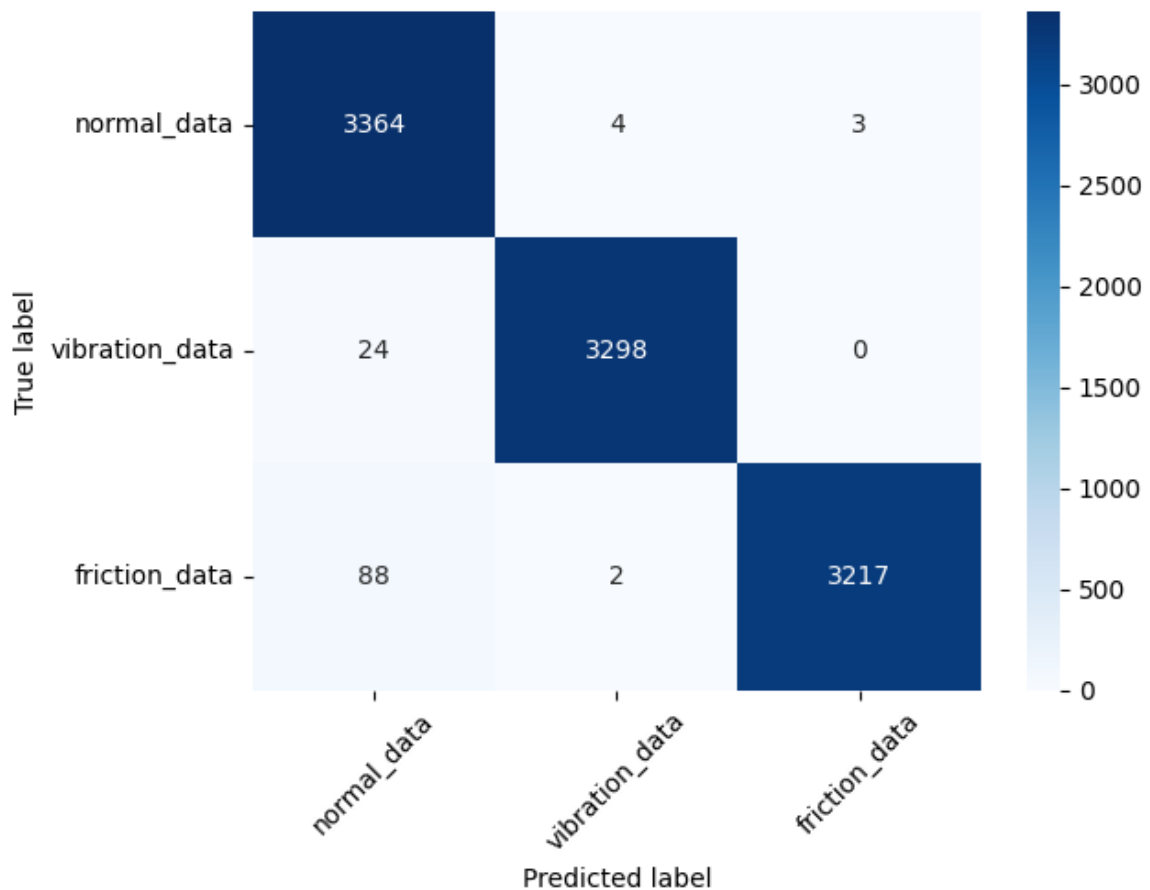


Figure 20: LSTM Classifier ([N,V,F], 20 epochs, lr0.001, seqLen400, batchSize40, nhidden256, nlayer3)

Taking the LSTM classifier further, five datasets are used; N, V, F, RU and RD, depicted in Figure 21. The Confusion Matrix indicates that the LSTM is able to handle all datasets very well. There are some data in the RU and RD which is not classified as well as the other datasets, but overall, the LSTM classifier is doing a better job than the CWT classifier.

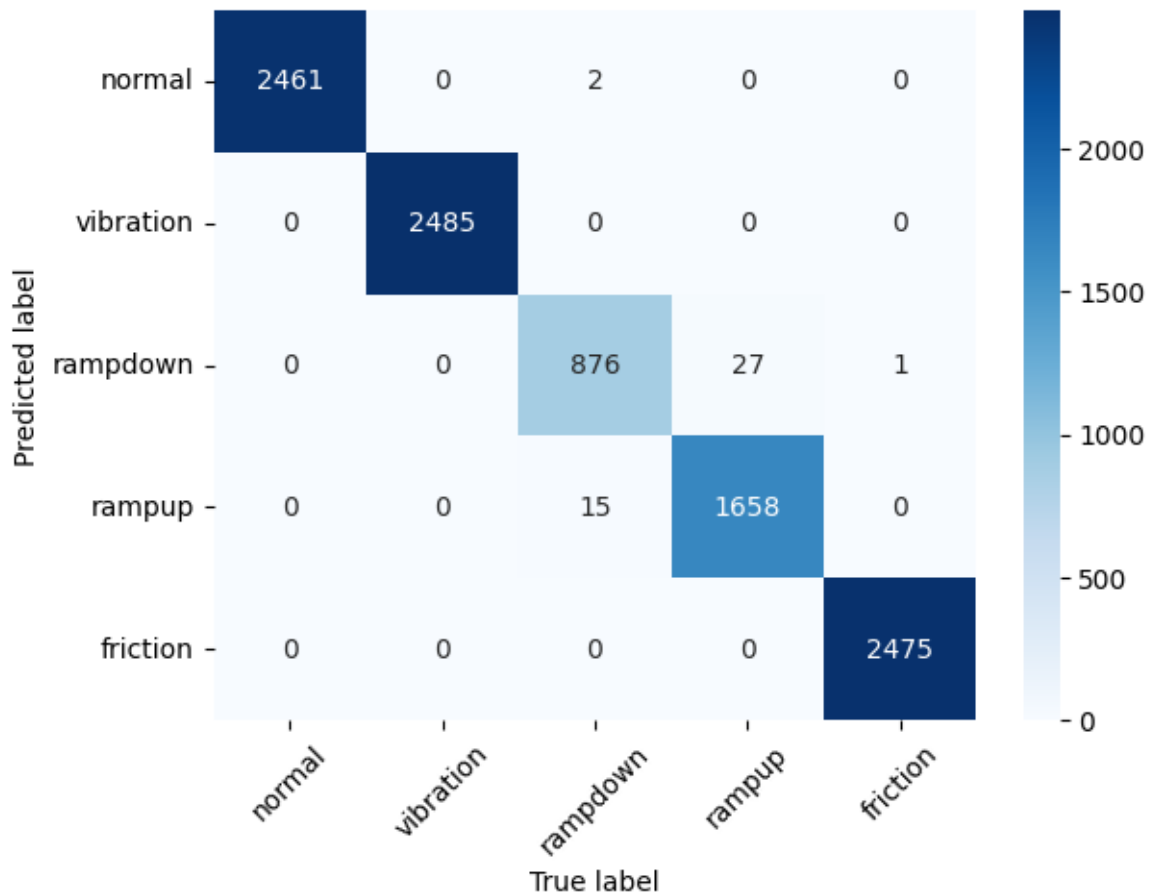


Figure 21: LSTM Classifier ([N,V,F,RU,RD], 20 epochs, lr0.001, seqLen400, batchSize40, nhidden256, nlayer3)

## 5.3 Clustering

The clustering results show how the clustering methods were implemented, and the general results of the clustering on the test sets.

### 5.3.1 Clustering using VGG19 and CTW

The clustering method using the pre-trained model VGG19 (4.4.4) and using CTW to generate scalograms (4.5.4) starts with the scalogram generation as shown in the process diagram in Figure 22. As there are no training phase in the clustering method, the next step only uses the test scalograms used in CWT – Classification (5.2.1). The next step takes the VGG19 model and removes the output layers (4.4.4). From there the scalograms are run through the stripped version of the VGG19 model, and the features are extracted. The next step is to reduce the dimensions of the features with PCA (4.4.3). Then the data is clustered using K-Means (4.4.1). Finally, the metrics are generated from the result of the clustering.

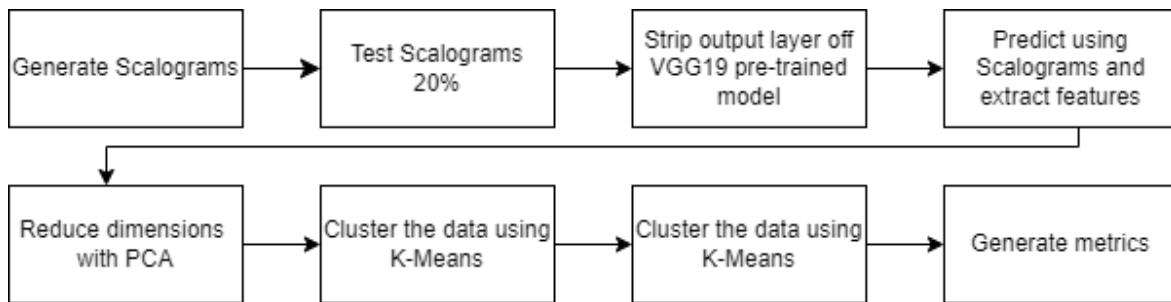


Figure 22: Overview diagram of clustering using VGG19 and CWT process

The first result shown in Figure 23 uses N, V, and F. From the figure, it's clear that each dataset is clustered into three distinct classes. This demonstrates effective clustering, indicating that the CWT method works well for clustering these datasets.

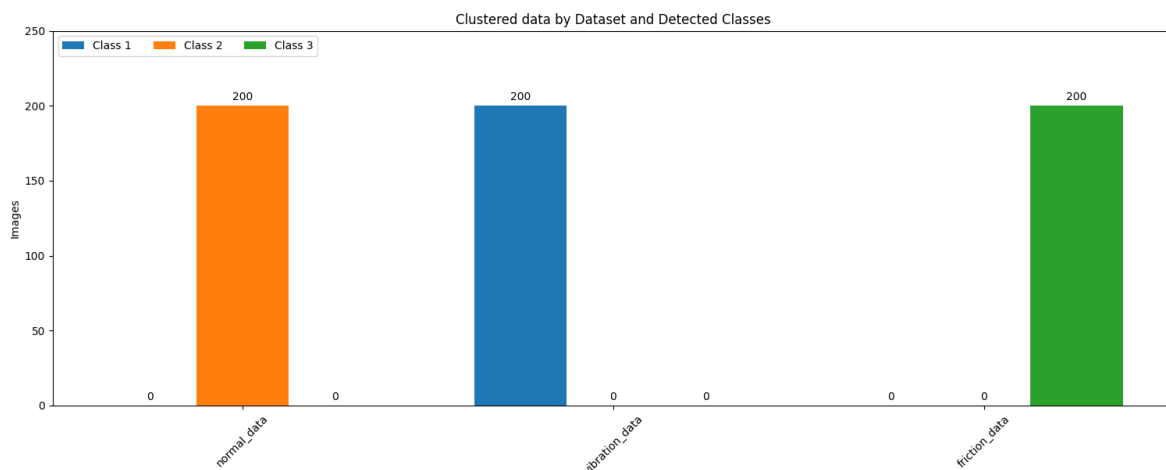


Figure 23: CWT Clustering ([N,V,F], seqLen 120, PCA 10)

The next result, shown in Figure 23, uses the datasets N, F, V, RU, and RD. The results indicate that while CWT clusters N, F, and V effectively, it struggles with the RU and RD datasets. This trend is also observed in the CWT classifier. The results clearly show that both RU and RD are getting mixed with N, represented in blue.

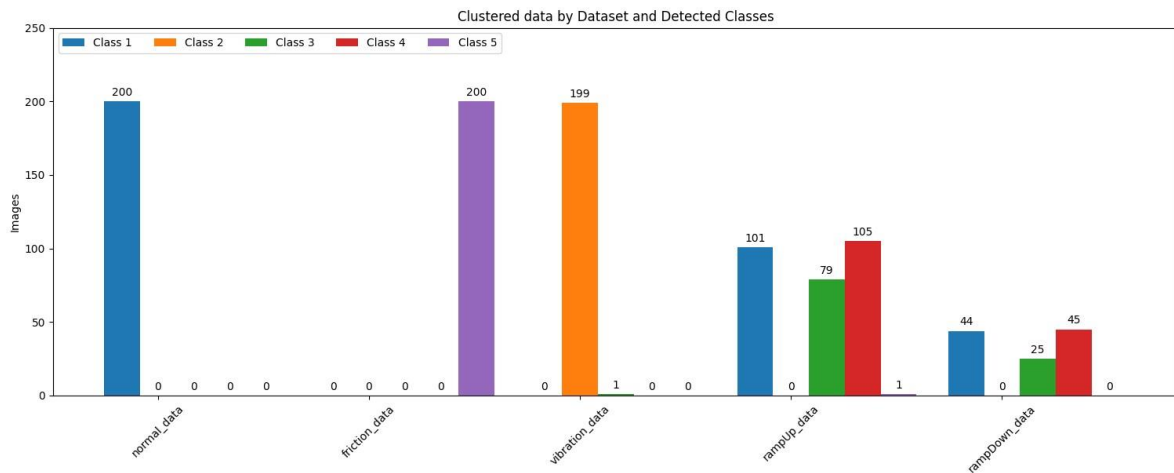


Figure 24: CWT Clustering ([N,V,F,RU,RD], seqlen 120, PCA 10)

### 5.3.2 DTW + K-Medoids

The clustering process using K-Medoids (4.4.2) and DWT (4.1) is illustrated in Figure 25. It begins with the generation of sequences (4.5.1), which are then clustered using K-Medoids, with DTW as the distance measurement. Metrics are generated based on the created clusters.

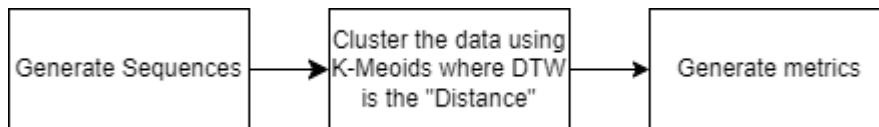


Figure 25: Overview diagram of clustering using K-Medoids and DWT process

The first result, depicted in Figure 26, shows the K-Medoids clustering applied to the N, V, and F datasets. It's clear that all datasets are detected well, although V and F miss some predictions and are not clustered as effectively as N.

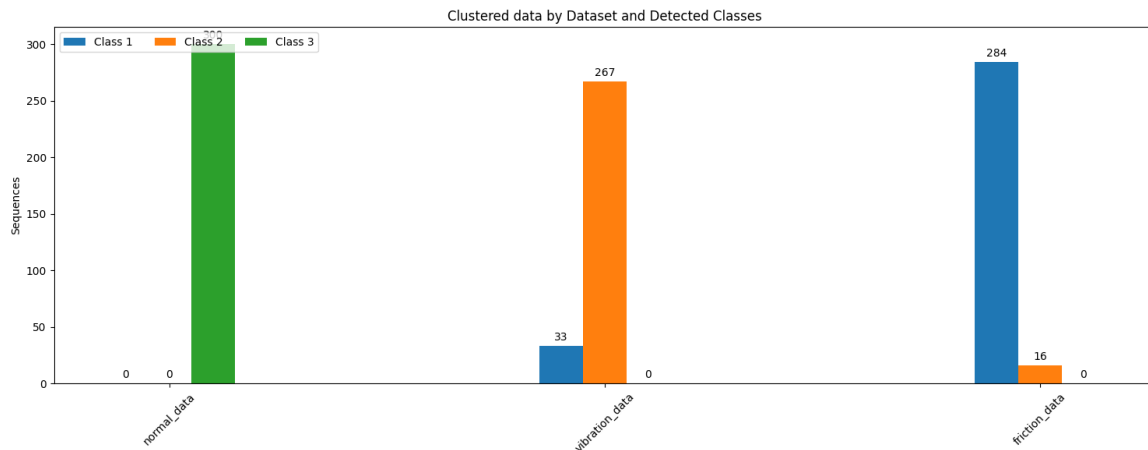


Figure 26: DTW and K-Medoids clustering ([N,V,F], seqlen 400, maxiter1000)

Pushing the K-Medoids with DTW further using five variables, as shown in Figure 27, reveals that the method struggles. No classes are clearly separated. While it might be possible to distinguish N as class 5 (purple), even this classification has uncertainties.

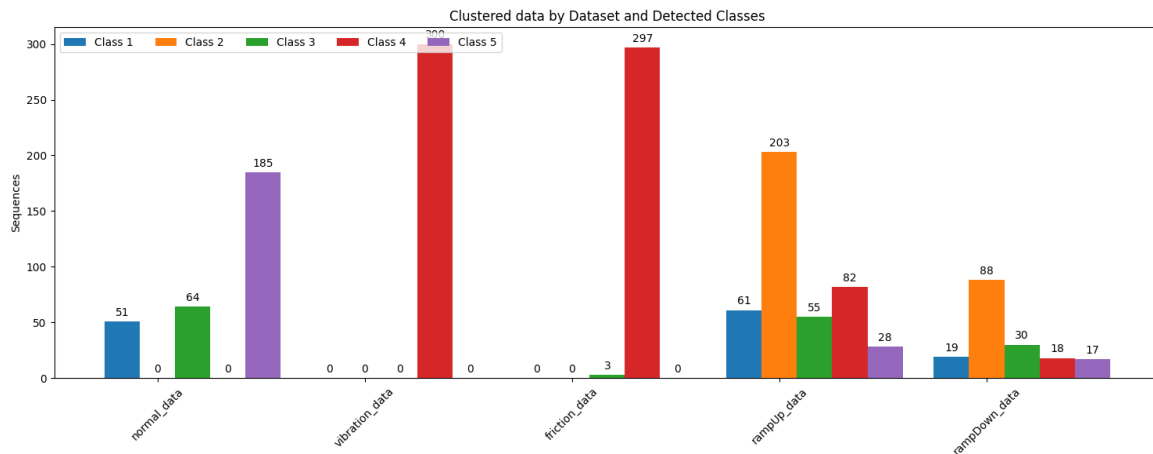


Figure 27: DTW and K-Medoids clustering ([N,V,F, RU, RD], seqlen 600, maxiter3000)

## 5.4 Comparison

To get a good understanding of the difference between the methods, chapter goes through comparing the classifiers, the clustering methods and then finally comparing the clustering to the classification methods.

### 5.4.1 Classification comparison

The first comparison involves N, V, and F, as shown in Figure 28. The results indicate that both models perform well on these datasets. However, the CWT model stands out as the better one, with no errors, though both models are overall quite effective.

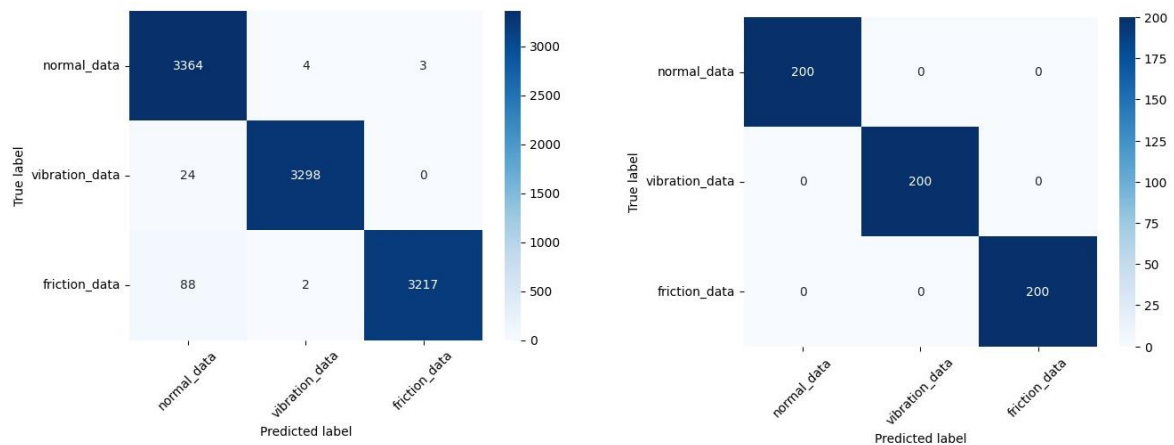


Figure 28: Comparison of classification methods LSTM (left) and CWT (right) comparing [N, V, F]

The next comparison, depicted in Figure 29, involves N, VL, and FL, which are intended to be challenging to detect. This comparison reveals a stark contrast between the models. The LSTM model outperforms the CWT model, which struggles to make accurate predictions for VL and FL. On a positive note, the CWT model can still detect the normal dataset, indicating some level of differentiation.

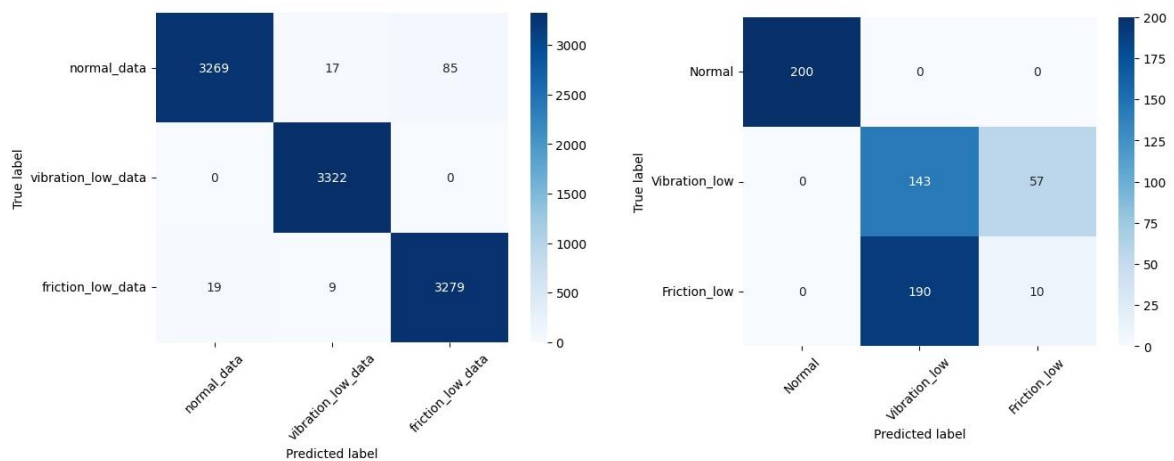


Figure 29: Comparison of classification methods LSTM (left) and CWT (right) comparing [N, VL, FL]

The final direct comparison, shown in Figure 30., involves the N, V, F, RU, and RD datasets. Once again, the LSTM model outperforms the CWT classifier. While the CWT model does a good job classifying N, V, and F, it struggles with RU and RD. The LSTM model performs better in identifying RU and RD, though these remain the two weakest datasets for the LSTM model in terms of correct predictions.



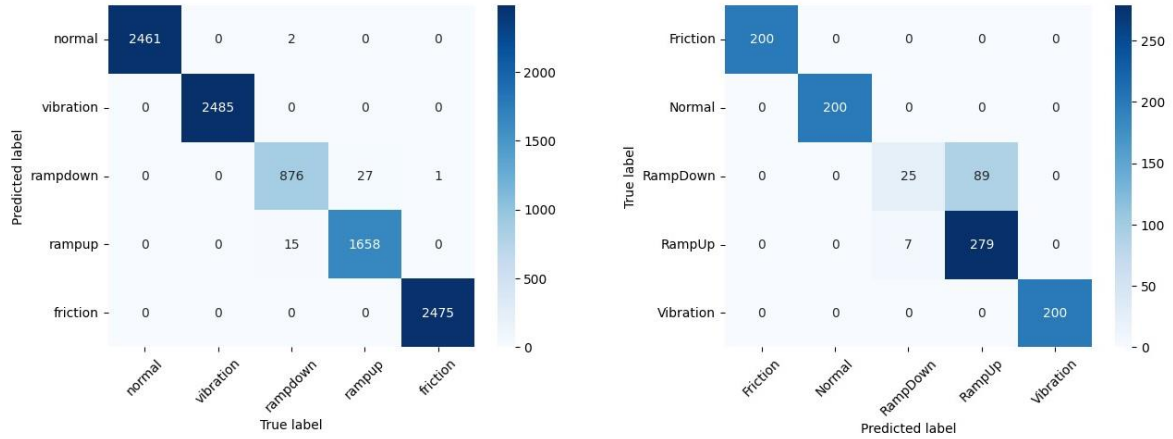


Figure 30: Comparison of classification methods LSTM (left) and CWT (right) comparing [N, V, F, RU, RD]

Finally, comparing the metrics in Table 9 between the LSTM model and the CWT model, it is clear that overall, the LSTM model performs better. The LSTM metrics are all close to one across the different dataset combinations, whereas the CWT model struggles with VL, FL, RU, and RD. However, it's important to note the difference in training time: the CWT model takes between 0-2 hours for all combinations, while the LSTM model requires around 8-15 hours, depending on the number of epochs needed.

Table 9: Classification metrics

	<i>Epochs</i>	<i>Recall</i>	<i>Precision</i>	<i>F1 Score</i>	<i>Accuracy</i>
<i>3 - CWT [N, V, F]</i>	15	1.00	1.00	1.00	1.00
<i>3 - LSTM [N, V, F]</i>	20	0.99	0.99	0.99	0.99
<i>3 - CWT [N, VL, FL]</i>	15	0.53	0.59	0.54	0.59
<i>3 - LSTM [N, VL, FL]</i>	15	0.97	0.97	0.97	0.97
<i>5 - CWT [N, V, F, RU, RD]</i>	15	0.91	0.90	0.88	0.90
<i>5 - LSTM [N, V, F, RU, RD]</i>	10	1.00	1.00	1.00	1.00

In summary, the LSTM model performs better overall but takes significantly longer to train compared to the CWT model. While the LSTM model performs worse than the CWT model on simpler tasks, such as the N, V, F combination, it excels with more complex datasets.

### 5.4.2 Clustering comparison

The first clustering comparison, shown in Figure 31, demonstrates that both models perform well in clustering the data. The CWT model is perfect, while the DTW-based model lags slightly behind.

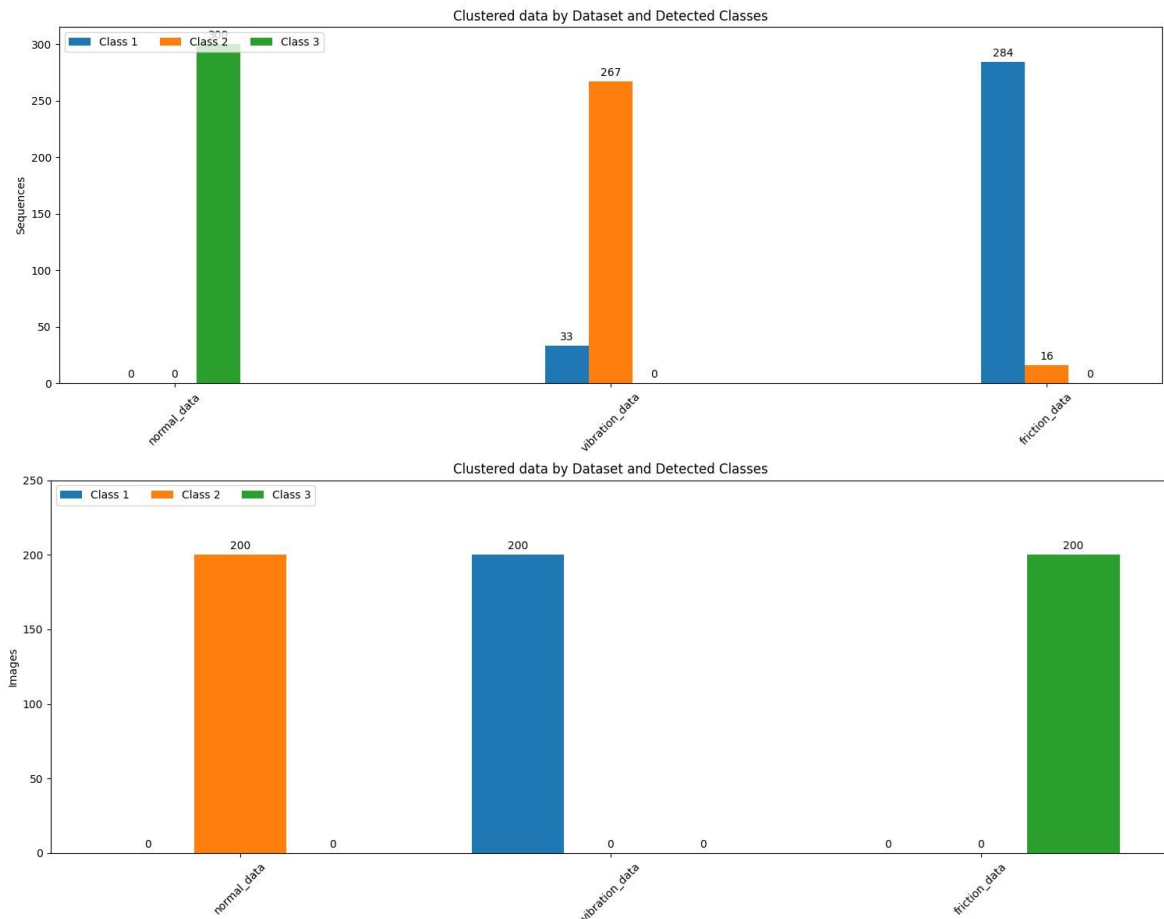


Figure 31: Comparison of clustering methods DTW (top) and CWT (bottom) comparing [N, V, F]

Extending the comparison in Figure 32, N, VL, and FL are analyzed. Both models struggle to cluster these datasets effectively, identifying similarities in different ways. The DTW-based model groups VL and FL together as one class, while the CWT-based model spreads two classes out over two datasets.

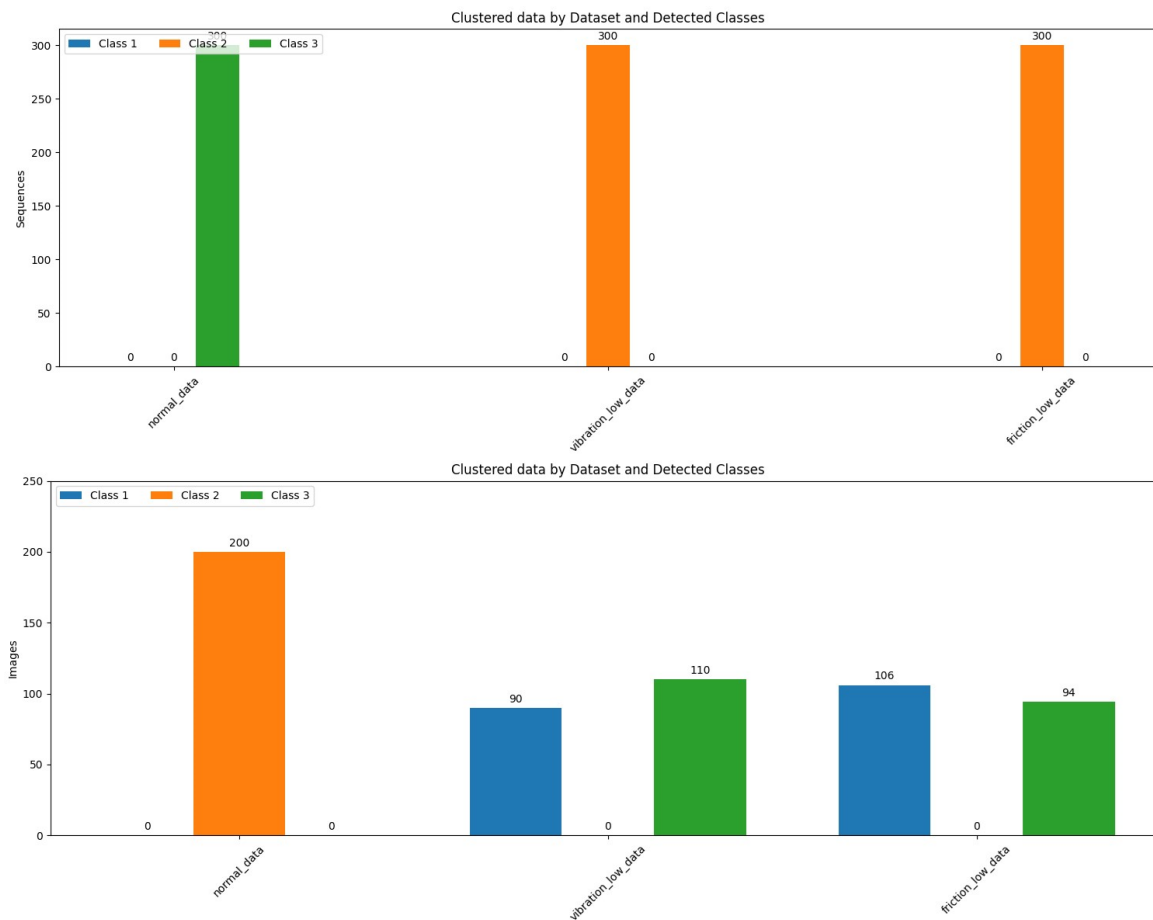


Figure 32: Comparison of clustering methods DTW (top) and CWT (bottom) comparing [N, VL, FL]

The final comparison, shown in Figure 33, reveals that both methods struggle. The DTW-based method creates only two clusters, with N being the only dataset classified correctly. The CWT-based method manages to find three clusters but fails to differentiate between RU and RD.

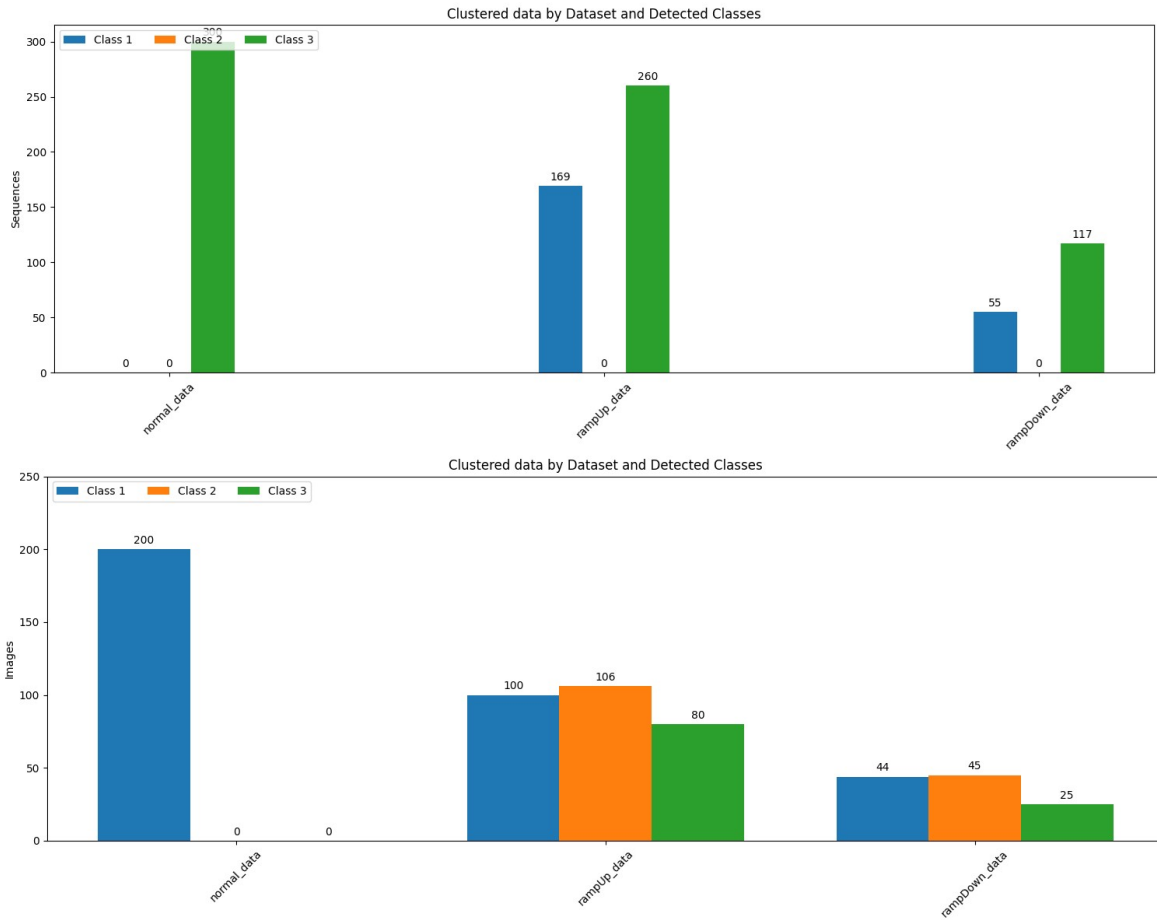


Figure 33: Comparison of clustering methods DTW (top) and CWT (bottom) comparing [N, RU, RD]

Overall, the clustering methods both fail with VL, FL, RU, and RD, showing poor performance. However, for the more distinct datasets, N, V, and F, the performance is good, with the CWT-based model performing the best.

### 5.4.3 Real-world implementation

While the accuracy of various models is crucial, their performance post-training is equally important. For real-time motor detection systems, prediction speed must be sufficiently low. Although defining an exact threshold is challenging, a prediction speed of less than one second is desirable, according to the author and supervisors.

To assess this, performance measurements are conducted on the different models. While this comparison isn't highly scientific, as speeds will vary based on hardware, it provides a useful estimate of expected performance.

The results of these performance measurements are shown in Table 10. Overall, the performance of all the models falls within reasonable prediction times. DTW is by far the fastest but also the least effective model overall.

Table 10: Measured prediction times on models based on 3 datasets.

	<i>LSTM - Classification [ms]</i>	<i>CWT - Classification [ms]</i>	<i>CWT - Clustering[ms]</i>	<i>DTW - Clustering[ms]</i>
<i>Average Time</i>	226.6	246.6	119.0	5.6
<i>Highest</i>	252.0	261.0	276.8	7
<i>Lowest</i>	42.1	169.0	110.9	4.9

## 6 Discussion

This chapter discusses the key findings of the thesis, "Clustering Time-Series Data," and their implications. The study aimed to validate a test rig (2.1) for generating motor vibration data suitable for classification and clustering, comparing various methods (4) to understand their effectiveness. The discussion highlights the significance of these findings in the context of existing research and practical applications, with a focus on classification performance, clustering performance, methodological considerations, practical implications, limitations, and future work.

### 6.1 Classification Performance

The ResNet18 model combined with Continuous Wavelet Transform (CWT) showed high accuracy in classifying the N, V, and F datasets. However, the model struggled with the VL, FL, RU, and RD datasets. The minimal deviations in VL and FL from the N conditions made them challenging to classify, while the time-dependent characteristics of RU and RD further complicated the classification process.

The visual differences in scalograms (5.1), particularly the sporadic lines in the V dataset and the distinct patterns in the F dataset, was likely a contributor to the model's strong performance in the N, V and F datasets. The VL and FL datasets were visually not identifiable. This indicates that a visual difference in the scalograms can be used as an early metric to understand how likely good classification is. More work should be put in to creating scalograms and visually show the difference when using CWT with ResNet18.

The results from chapter 5.1 underline important issue with the way the scalogram are generated, especially considering datasets like RU and RD. The scalograms generated from RU and RD show big variations in scalograms depending on where in the ramp-up or ramp-down phase of the scalograms are generated. This could be solved by extending the length of the scalograms (120 is the length used in the classification here), but then some of the real time monitoring capabilities are lost.

The Long Short-Term Memory (LSTM) model outperformed ResNet18 in handling more complex and less distinct datasets (5.2.2). The LSTM's ability to capture long-term dependencies in time-series data (4.3.1) proved advantageous, particularly with the RU and RD datasets. Although the LSTM model had a slightly lower accuracy for the simpler N, V, and F datasets, it excelled with the more challenging datasets. The trade-off between accuracy and training time was evident, with the LSTM requiring significantly longer training periods.

### 6.2 Clustering Performance

The VGG19 model combined with CWT for feature extraction showed excellent clustering performance for the distinct datasets N, V and F (5.3.1). The scalograms provided rich visual features that facilitated effective clustering. However, the model struggled with the VL, FL, RU, and RD datasets, similar to the classification challenges.

There appears to be similar performance in clustering and classification when using CWT with a pre-trained model. This could indicate that the method is working well, and more work into creating more distinct scalograms is needed to further these methods.

Overall, it's clear that the CWT methods perform best on repeating patterns that don't change over long periods of time.

The last method, Dynamic Time Warping (DTW) combined with K-Medoids also showed promising results for the N, V, and F datasets (5.3.2). DTW's ability to align sequences that vary in time or speed made it suitable for these datasets. However, its performance declined with the VL, FL, RU, and RD datasets, where the distinctions were subtler. It's unclear why the DTW and K-Medoids method performed poorly compared to the other methods.

### **6.3 Methodological Considerations**

Sequencing, normalization, and dataset balancing significantly impacted model performance. Sequencing was essential for creating manageable input sizes for the models, where an overlapping sequencing was used for the LSTM, and a non-overlapping sequencing was used for the other methods. Dataset balancing was not necessary, as all data was within acceptable range, as shown in 4.5.3.

### **6.4 Practical Implications**

The findings have significant implications for industries relying on motor-based systems. Improved classification and clustering methods can enhance Condition-Based Monitoring (CBM), reducing downtime and maintenance costs. The ability to accurately classify and cluster motor vibration data enables preemptive identification of potential issues, enhancing operational efficiency and system reliability.

The feasibility of implementing these models in real-time monitoring systems was also evaluated. The performance measurements indicated that all models could make predictions in under one second, suitable for real-time applications. The DTW model, despite its lower overall performance, was the fastest in prediction speed.

### **6.5 Limitations and Challenges**

The datasets used had limitations in size, diversity, and potential biases. The test rig setup, while controlled, may not fully represent all real-world conditions. Future studies should aim to implement real-time methods on the test rig and on industrial installations.

Each model had inherent limitations. ResNet18 and VGG19, while effective for visual data, struggled with subtle time-series distinctions. LSTM, although powerful for capturing long-term dependencies, required extensive computational resources and training time. DTW, despite its speed, had lower overall accuracy.

### **6.6 Future Work**

Future research should explore enhancements to the classification and clustering models, especially the CWT methods, where improvements to the scalograms could greatly improve the classification according to the results.

The methods developed could be extended to other industrial applications beyond motor vibration data. Exploring applications in areas such as predictive maintenance for other types of machinery or even in different domains like healthcare for monitoring vital signs could be valuable.



## 7 Conclusion

The findings of this thesis contribute significantly to the field of condition-based monitoring and time-series data analysis. The test rig is proven to generate vibration data suitable for classification, and the comparison of various classification and clustering methods provides valuable insights into their effectiveness and practical applications. While the LSTM showed superior performance in handling complex datasets, ResNet18 and VGG19 also demonstrated strong capabilities with more distinct data. The study highlights the importance of methodological considerations, practical implications, and potential areas for future research, aiming to improve the reliability and efficiency of condition-based monitoring systems.

## References

- [1] “DALL·E 3 Image Generation Based On Reference Image.” Accessed: May 15, 2024. [Online]. Available: <https://openai.com/index/dall-e-3/>
- [2] A. Ali and A. Abdelhadi, “Condition-Based Monitoring and Maintenance: State of the Art Review,” *Applied Sciences*, vol. 12, p. 688, May 2022, doi: 10.3390/app12020688.
- [3] J. Wang and C. Zhao, “Robust Control Performance Monitoring for Varying-Dimensional Time-Series Data Based on SCADA Systems,” *IEEE Trans Instrum Meas*, vol. 71, p. 1, May 2022, doi: 10.1109/TIM.2022.3177217.
- [4] “BEDKO AS | Intelligent systems.” Accessed: Apr. 23, 2024. [Online]. Available: <https://www.bedko.com/>
- [5] “MPU-9250 | TDK InvenSense.” Accessed: May 14, 2024. [Online]. Available: <https://invensense.tdk.com/products/motion-tracking/9-axis/mpu-9250/>
- [6] “InfluxDB Time Series Data Platform | InfluxData.” Accessed: Apr. 23, 2024. [Online]. Available: <https://www.influxdata.com/>
- [7] “InfluxDB | InfluxData.” Accessed: May 05, 2024. [Online]. Available: <https://www.influxdata.com/products/influxdb/>
- [8] E. Quatrini, F. Costantino, G. Di Gravio, and R. Patriarca, “Condition-Based Maintenance—An Extensive Literature Review,” *Machines*, vol. 8, p. 31, May 2020, doi: 10.3390/machines8020031.
- [9] U. Ilic, B. Trojic, V. Lazic, and F. Filipovic, “Classification models of machine learning for vibration analysis of induction motor,” May 2019.
- [10] S. Fong, J. Harmouche, S. Narasimhan, and J. Antoni, “Mean Shift Clustering-Based Analysis of Non-Stationary Vibration Signals for Machinery Diagnostics,” *IEEE Trans Instrum Meas*, vol. PP, p. 1, May 2019, doi: 10.1109/TIM.2019.2944503.
- [11] S. Mahe, “DETERMINING ELECTRIC MOTOR BEARING FAULTS THROUGH VIBRATION ANALYSIS,” 2022.
- [12] “ResNet-18 convolutional neural network - MATLAB resnet18 - MathWorks Nordic.” Accessed: May 14, 2024. [Online]. Available: <https://se.mathworks.com/help/deeplearning/ref/resnet18.html>
- [13] “Continuous Wavelet Transform and Scale-Based Analysis - MATLAB & Simulink - MathWorks Nordic.” Accessed: May 14, 2024. [Online]. Available: <https://se.mathworks.com/help/wavelet/gs/continuous-wavelet-transform-and-scale-based-analysis.html>
- [14] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. O’Reilly Media, Inc., 2019.
- [15] “VGG-19 convolutional neural network - MATLAB vgg19 - MathWorks Nordic.” Accessed: May 14, 2024. [Online]. Available: <https://se.mathworks.com/help/deeplearning/ref/vgg19.html>

- [16] H. Kamper, “Dynamic Time Warping.” Accessed: May 14, 2024. [Online]. Available: <https://www.kamperh.com/slides/dtw-crop.pdf>
- [17] E. D. Ann Maharaj Pierpaolo and U. Jorge Caiado, “Time Series Clustering and Classification,” New York, 2019. doi: 10.1201/9780429058264.
- [18] D. Jordan, R. Miksad, and E. Powers, “Implementation of the continuous wavelet transform for digital time series analysis,” *Review of Scientific Instruments*, vol. 68, pp. 1484–1494, 1997, doi: 10.1063/1.1147636.
- [19] “Practical Introduction to Time-Frequency Analysis Using the Continuous Wavelet Transform - MATLAB & Simulink Example - MathWorks Nordic.” Accessed: May 14, 2024. [Online]. Available: <https://se.mathworks.com/help/wavelet/ug/practical-introduction-to-time-frequency-analysis-using-the-continuous-wavelet-transform.html>
- [20] D. Barache, J. Antoine, and J. Dereppe, “The continuous wavelet transform, an analysis tool for NMR spectroscopy,” *Journal of Magnetic Resonance*, vol. 128, pp. 1–11, 1997, doi: 10.1006/JMRE.1997.1214.
- [21] P. Kumar and E. Foufoula-Georgiou, “Wavelet analysis for geophysical applications,” *Reviews of Geophysics*, vol. 35, pp. 385–412, 1997, doi: 10.1029/97RG00427.
- [22] I. Legarreta, P. Addison, N. Grubb, G. Clegg, C. Robertson, and J. N. Watson, “A comparison of continuous wavelet transform and modulus maxima analysis of characteristic ECG features,” *Computers in Cardiology, 2005*, pp. 755–758, 2005, doi: 10.1109/CIC.2005.1588214.
- [23] J. B. MacQueen, “Some Methods for Classification and Analysis of MultiVariate Observations,” in *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. Le Cam and J. Neyman, Eds., University of California Press, 1967, pp. 281–297.
- [24] T. Kodinariya and P. Makwana, “Review on Determining of Cluster in K-means Clustering,” *International Journal of Advance Research in Computer Science and Management Studies*, vol. 1, pp. 90–95, Apr. 2013.
- [25] M. E. Celebi, H. Kingravi, and P. Vela, “A Comparative Study of Efficient Initialization Methods for the K-Means Clustering Algorithm,” *Expert Syst Appl*, vol. 40, pp. 200–210, Apr. 2013, doi: 10.1016/j.eswa.2012.07.021.
- [26] V. Niennattrakul and C. Ratanamahatana, “On Clustering Multimedia Time Series Data Using K-Means and Dynamic Time Warping,” in *Multimedia and Ubiquitous Engineering International Conference*, Apr. 2007, pp. 733–738. doi: 10.1109/MUE.2007.165.
- [27] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction To Cluster Analysis*. 1990. doi: 10.2307/2532178.
- [28] F. L. Gewers *et al.*, “Principal Component Analysis,” *ACM Computing Surveys (CSUR)*, vol. 54, pp. 1–34, 2018, doi: 10.1145/3447755.
- [29] G. Lorenz, “Principal Component Analysis in Technology,” *CIRP Annals*, vol. 38, pp. 107–109, 1989, doi: 10.1016/S0007-8506(07)62662-6.
- [30] R. Bro and A. Smilde, “Principal component analysis,” *Analytical Methods*, vol. 6, pp. 2812–2831, 2014, doi: 10.1039/C3AY41907J.

- [31] S. Coggeshall and G. Wu, “Asset Allocation and Long-Term Returns: An Empirical Approach,” *SSRN Electronic Journal*, May 2005, doi: 10.2139/ssrn.873184.
- [32] D. Singh and B. Singh, “Investigating the impact of data normalization on classification performance,” *Appl Soft Comput*, vol. 97, p. 105524, Dec. 2020, doi: 10.1016/j.asoc.2019.105524.
- [33] V. V. Starovoitov and Yu. I. Golub, “Data normalization in machine learning,” *Informatics*, vol. 18, no. 3, pp. 83–96, Sep. 2021, doi: 10.37661/1816-0301-2021-18-3-83-96.
- [34] S. Ali and K. A. Smith-Miles, “Improved Support Vector Machine Generalization Using Normalized Input Space,” 2006, pp. 362–371. doi: 10.1007/11941439\_40.
- [35] Y. Wu and K. He, “Group Normalization,” *Int J Comput Vis*, vol. 128, no. 3, pp. 742–755, Mar. 2020, doi: 10.1007/s11263-019-01198-w.
- [36] “Imbalanced Data | Machine Learning | Google for Developers.” Accessed: May 15, 2024. [Online]. Available: <https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data>
- [37] “lecture\_dtw\_notebook/dtw.ipynb at main · kamperh/lecture\_dtw\_notebook · GitHub.” Accessed: Apr. 24, 2024. [Online]. Available: [https://github.com/kamperh/lecture\\_dtw\\_notebook/blob/main/dtw.ipynb](https://github.com/kamperh/lecture_dtw_notebook/blob/main/dtw.ipynb)

# Appendices

Appendix A – Main Code files, results, and data

<https://github.com/zovz/MTAR>